NTNU – Trondheim
Norwegian University of
Science and Technology

Operational Planning and Disruption Management in Offshore Logistics

Nils Blegeberg Albjerk
Teodor Kristoffer Danielsen
Stian Krey

Industrial Economics and Technology Management
Submission date: June 2015
Supervisor: Kjetil Fagerholt, IØT
Co-supervisor: Magnus Stålhane, IØT

Norwegian University of Science and Technology
Department of Industrial Economics and Technology Management

Problem description from the master's thesis agreement

The purpose of this thesis is to develop a mathematical model and different solution methods for handling disruptions in operational planning in offshore logistics. The problem is based on a real setting faced by an oil and gas producing company. A solution approach for the problem should balance costs associated with offshore supply and service level at offshore installations. Exact and heuristic solution approaches will be developed and compared to each other.

Main contents:

1. Description of the problem
2. Presentation of mathematical model
3. Implementation of mathematical model
4. Computational study of the model with realistic data

Cover photo: Nils B. Albjerk (2015)

Preface

This master's thesis is the result of the final work for our MSc. in Industrial economics and technology management, with specialization in managerial economics and operations research. The thesis is a continuation of the work done by Teodor N. Danielsen on his specialization project in the fall semester of 2014.

The thesis focuses on operational planning and disruption management in Statoil's offshore supply service on the Norwegian continental shelf. The work is conducted as part of the MOLO (Maritime offshore logistics optimization) project, which is a collaboration between Statoil, the Norwegian University of Science and Technolgy, the Norwegian Marine Technology Research Institute, and Molde University College.

We would like to thank our supervisors Professor Kjetil Fagerholt and Postdoctoral Fellow Magnus Stålhane for excellent guidance and support in the project. We would also like to express our gratitude to supply chain management consultant Tor Toftøy in Statoil, who has contributed with valuable advice and information. Finally, we would like to thank the Statoil employees at the supply base in Kristiansund and the crew at the offshore supply vessel Olympic Energy who made it possible for us to experience the offshore supply operations in practice.

Trondheim, June 4th

Nils Blegeberg Albjerk Teodor Nilseng Danielsen Stian Krey

Abstract

This thesis considers operational planning and disruption management in the offshore supply service in Statoil, the biggest Norwegian oil and gas company. A significant amount of time is needed for operational planning, and major costs are caused by disruptions to the planned routes and schedules for the offshore supply vessels (OSVs) supplying the offshore installations. The disruptions are mainly due to uncertain and harsh weather conditions, unexpected orders placed by offshore installations, and uncertain order volumes. A decision tool based on mathematical models may contribute to reducing the costs and time related to planning and disruption management in the offshore supply service.

Mathematical models handling disruptions and finding new routes and schedules for OSVs are presented. The problem is modelled as a pickup and delivery problem with time limits. Exact solution methods are introduced, and due to the complexity of the problem, a heuristic is developed to solve realistic instances that the exact models are not able to solve within a reasonable amount of time.

The problem instances are based on data provided by Statoil. The computational study compares the solutions of the proposed models. The results from the computational study show that the heuristic finds optimal solutions for all the problem instances where optimality can be proven by the exact methods, and has a stable performance for the other larger instances. The heuristic can form the basis for a decision support tool that can be utilized in everyday operational planning and disruption management by Statoil. Applying the heuristic for decision support can reduce the time spent on planning, and significant cost reductions can be made in the offshore supply service.

Sammendrag

Denne oppgaven handler om operasjonell planlegging og avvikshåndtering i forbindelse med betjening av Statoils offshoreinstallasjoner. Operasjonell planlegging krever mye tid, og avvik fra forsyningsskipenes planlagte ruter medfører store kostnader i verdikjeden. Avvikene oppstår i hovedsak på grunn av usikre og tidvis utfordrende værforhold, uventede ordre fra installasjoner og usikker størrelse på ordre. Et beslutningsstøtteverktøy som er basert på matematiske modeller kan bidra til å redusere kostnadene og tidsbruken forbundet med planlegging og avvikshåndtering knyttet til betjeningen av offshoreinstallasjonene.

Matematiske modeller som håndterer avvik og finner nye ruter for forsyningsskipene er presentert. Eksakte løsningsmetoder er utviklet, og grunnet kompleksiteten til problemene, presenteres det også en heuristikk som er i stand til å løse større problemer enn de eksakte modellene klarer innen rimelig tid.

Probleminstansene er basert på data fra Statoil. I oppgavens resultatstudie, er løsningene til de foreslåtte modellene sammenlignet. Resultatstudiet viser at heuristikken finner optimal løsning for alle problemene der optimalitet kan bevises av de eksakte metodene, og at den utover dette har en stabil ytelse for de større problemene. Heuristikken kan danne grunnlaget for et beslutningsstøtteverktøy som kan brukes på daglig basis i operasjonell planlegging og avvikshåndtering for betjening av olje- og gassinstallasjoner offshore. Et slikt verktøy kan redusere tiden som brukes på planlegging og kostnadene knyttet til betjening av offshoreinstallasjonene.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Oil and gas production on the Norwegian continental shelf began more than four decades ago. With the discovery of the field Ekofisk, a series of large findings in the North Sea was initiated. Increasing knowledge and new technology has gradually moved production north to the Norwegian Sea and the Barents Sea as well. The North Sea is still the main production area with a total of 60 active oil and gas fields. The large oil and gas reserves on the Norwegian continental shelf have turned Norway into one of the biggest oil and gas producers in the world. The industry has generated more than NOK 11 000 billion in present value since the first discoveries. Although there has been a continuous production for over 40 years, it is estimated that 56% of the total resources have still not been extracted (Regjeringen, 2014).

Even though close to a third of all value creation in Norway still happens in the petroleum sector, there is an increased focus on cost reduction in the industry, especially because of the drop in oil prices in the last half of 2014. In the survey DNV GL (2015), 47% of the respondents, mainly oil and gas operators, suppliers, and service companies, expect to reduce their headcount in 2015. However, other measures, such as new improved technology and increased efficiency in production are preferred, rather than reducing the number of employees.

Figure 1.1: The floating platform for gas production, Åsgard B, seen from an offshore supply vessel (OSV). Photo: Nils B. Albjerk, taken during an excursion on board the OSV Olympic Energy in the Norwegian sea visiting Statoil's offshore installations in February 2015.

Oil and gas production does not only generate vast revenues, it also requires bigger investments than most other industries. This is especially becoming a challenge as future oil and gas fields are expected to be smaller, the prices to be lower than the average over the last few years, and the resources less available than in previous years. At the same time, it is more costly to extract the remaining resources in existing fields, since the most available reserves have already been utilized. In other words, the petroleum industry is challenged by the need for more effective production at less available fields, in addition to necessary cost reductions.

The petroleum industry on the Norwegian continental shelf is regulated by the government in order to let value creation benefit the society. In this context, the state owned company Statoil was founded in 1972. Statoil is today the leading operator on the Norwegian continental shelf, and had a production of 1 927 million barrels of oil equivalents per day in 2014 (Statoil, 2015a). Like the rest of the Norwegian petroleum industry, the company is working to reduce costs in their value chain. This report will focus on efficiency challenges in Statoil's upstream supply chain with regard to supplying the offshore installations.

Statoil's production of oil and gas is performed at both fixed and floating offshore installations at fields in the North Sea, the Norwegian Sea, and the Barents Sea. Activities at offshore installations include drilling, well operations, and mainte-

Figure 1.2: An offshore supply vessel sailing in harsh weather conditions. Photo: Karlsson (2013)

nance. Depending on what kind of operations that are performed at an installation, certain supplies are needed from an onshore supply depot. Each offshore installation must identify its demand for different supplies in order to keep production, maintenance, and other activities running.

Supplies are brought to the installations by offshore supply vessels (OSVs), which are ships especially designed for this purpose. The OSVs are able to transport containers and other goods on deck, in addition to bulk cargo, which is kept in tanks below deck. The vessels also carry backload from the installations to the onshore supply depot. The OSVs are constructed to handle extreme weather conditions, which they often encounter on the Norwegian continental shelf. An OSV sailing in harsh weather conditions is shown in Figure 1.2.

Costs related to chartering and operating OSVs are one of the major expenses in the upstream supply chain. In addition, planning of routes and schedules for the vessels demands significant amounts of time and effort. The planning activities are performed at strategic, tactical, and operational level to decide optimal fleet size, optimal routes and schedules, and necessary adjustments to be made when unexpected demand is reported or delays occur.

This report focuses on operational planning and how disruptions to planned routes and schedules for OSVs can be handled. Disruptions are mainly due to adverse weather conditions reducing the ability of OSVs to sail, and making loading operations at offshore installations difficult. In addition to the weather conditions, unexpected orders are often reported by installations. These orders are in some cases a result of unforeseen events, but may also be placed late in the supply process because of insufficient planning at the offshore installations. Another factor leading to changes in planned routes and schedules is that the volume of orders is uncertain.

Deciding an optimal fleet, and planning of routes and schedules are done on a strategic and tactical level, while this report will focus on challenges occurring at the operational level. The report contributes to the development of decision tools to be used for OSV planning in an operational setting. The main purpose is to present how decisions may be based on mathematical models finding routes and schedules, and reducing the negative effects of delays and unexpected orders. Most existing literature studying the offshore upstream supply chain presents strategic and tactical problems, and does not provide solutions for making operational decisions.

In Chapter 2, the background for the problem discussed in this report is presented. The chapter starts with a description of the upstream supply chain for offshore oil and gas production, followed by an explanation of some of the planning processes. In Chapter 3, literature relevant to the problem is discussed, and in Chapter 4, a detailed problem description is given. In Chapter 5, a mathematical formulation of the problem is presented. An improved exact solution method, involving pregeneration of voyages through dynamic programming and a voyage-based formulation of the problem, is presented in Chapter 6. In Chapter 7, a variable neighborhood search heuristic is introduced. Chapter 8 contains the computational study, where the problem instances are presented, cost parameters are chosen, heuristic parameters are tuned, solution methods are compared, and some of the solutions are analyzed. The report is concluded in Chapter 9.

Chapter 2

Background

This chapter gives an overview of the offshore logistics operations performed on the Norwegian continental shelf. In Section 2.1, the upstream supply chain for offshore oil and gas production is presented, followed by a description of offshore logistics in Section 2.2. The planning processes needed for a reliable and cost effective offshore supply service is presented in Section 2.3.

2.1 The upstream supply chain

The main activities in the overall supply chain for petroleum production are shown in Figure 2.1. As this report focuses on logistics in the upstream supply chain, the downstream part will not be considered here.

The first step in the upstream supply chain involves exploration of new possible reserves. By exploring the subsurface through seismic, electromagnetism, gravitation, and magnetism, geophysicists are able to get a detailed understanding of the structures underneath the surface. Due to the high levels of risk and uncertainty in development of new oil and gas fields, extensive knowledge about the new reservoirs is essential for profitable production.

Drilling of wells is the only possible way of guaranteeing the existence of oil and

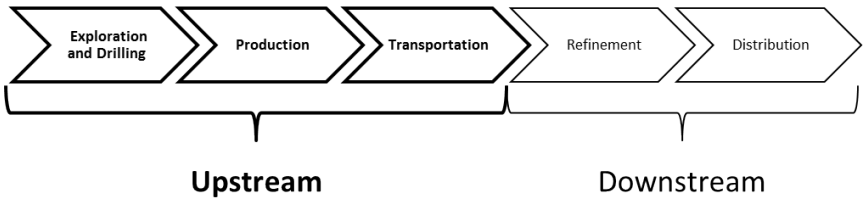

Figure 2.1: Main activities in the supply chain for offshore petroleum production.

gas in possible reservoirs. Since drilling involves large expenses, big investments are made to improve the technology used to identify the properties of possible oil and gas reservoirs before wells are drilled. In addition, there is emphasis on using drilling technologies that have low environmental impact and that are cost effective.

The importance of finding improved solutions for developing existing fields is increasing as oil and gas reserves become less available. Also, the predicted increased energy demand in the future requires more efficient extraction of the resources.

Among several oil and gas producing companies, Statoil is the leading operator on the Norwegian continental shelf (Statoil, 2015a). Offshore oil and gas production provide vast monetary values, thus reliable deliveries of supplies critical to production are necessary. Offshore logistics are further described in the next section. There are different kinds of offshore installations, such as:

- Production ships
- Floating production platforms
- Storage ships
- Fixed platforms
- Drilling rigs

Wells are drilled using drilling rigs that are leased by Statoil. When a well is ready for production, a production ship or platform takes over the operations.

Figure 2.2: A storage ship for condensate, Åsgard C, seen from an OSV. Photo: Nils B. Albjerk, February 2015.

The extracted oil and gas are brought to onshore storages. Gas produced on the Norwegian continental shelf is transported through the world's largest subsea gas pipeline network to mainland in Norway, Britain, Germany, Belgium, and France. The gas is transported directly in the subsea network, while the oil and condensates are either sent through pipes to storage ships or to the mainland.

The mentioned activities form the basis for many companies producing offshore installations, equipment and machinery, logistics companies, and companies specialized on searching for fields and on drilling. To illustrate the size of the petroleum supplier industry; the annual value of procurements in Statoil was more than NOK 170 billion in 2013 (Statoil, 2014b).

2.2 Offshore logistics

The oil and gas production on offshore installations depends on several types of supplies. Most of these are provided by OSVs. Statoil has currently 34 fixed and about 15 floating installations on the Norwegian continental shelf (Statoil, 2015b). According to Statoil employees, these are serviced by approximately 25 OSVs. The

only way staff or other personnel are transported to offshore installations is by helicopter. Helicopters are also used to transport goods in case of especially urgent deliveries. The process of delivering supplies to offshore installations is illustrated in Figure 2.3.

Due to high activity offshore, there is a continuous flow of demand for materials and products reported by the installations. The demanded deliveries are either kept in stock at the onshore supply depot or ordered from external suppliers. All materials must be properly packed and secured before entering the depot. The security requirements are strict, and packages from external suppliers must be repacked at the supply base if the supplier is not approved by the Norwegian Oil and Gas Association, which is an interest organization for the suppliers on the Norwegian continental shelf (The Norwegian Oil Gas Association, 2015). At the supply base, the materials are assigned to OSVs and further prepared for departure. Because of environmental impacts, Statoil has a goal of getting most deliveries from suppliers by sea, instead of by road transportation. However, the costs of hiring equipment needed at the offshore installations are high (often >NOK 500 000/day), and transportation by ship is time consuming. Therefore, most of the hired equipment is still carried by trucks on mainland.

The so-called *hold principle* is implemented at the onshore supply depots. This means that supplies are kept at the depot until short time before they are needed at the offshore installations. This is a measure towards keeping the amount of stored goods at installations at a minimal level, as storage capacity is a limited resource. Statoil is working on having the same principle implemented at their external suppliers in order to reduce the stock at the onshore supply depots. Even though Statoil tries to keep the stock levels at the supply bases down, they also need to have essential mechanical parts at hand at any time. The reason for this is that a shut down on an installation caused by one single part might cause huge economical losses (>NOK 200 000/hour). Since the time of delivery from supplier to Statoil might range from 2-12 months, the depot is required to have huge warehouses. The

Figure 2.3: The process of supplying offshore installations. Illustration: Statoil (2015c).

supply base in Kristiansund services nine installations and keeps a stock of 74 000 items worth more than NOK 1 billion. The items in stock are continuously counted and maintained during the year, ensuring control of the warehouse inventory.

The supplies delivered to the installations are classified by the following types (Statoil, 2014a):

- Operation materials
- Bulk products
- Oil country tubular goods
- Project materials
- Drilling and well materials
- Maintenance and modification materials

Specialized vessels are used to transport what is known as oil country tubular goods (OCTG), which is mainly different kinds of tubes used for drilling of wells and for production. Not all depots supply these kind of products. The bulk products transported by the OSVs are mostly barite, brine, drill water, and drilling mud used in the drilling process, in addition to cement for the wells, diesel used as fuel, methanol for the pipes, and freshwater for personnel (Statoil, 2015d). The bulk products are transported by OSVs in tanks below deck, pipes are stacked directly on deck in bundles, while all other supplies are kept in containers of different sizes on deck, as illustrated in Figure 2.4.

When the OSVs arrive at offshore installations, the demanded products are loaded onto the installations. Bulk products are transferred from the vessel to the installation using a pipe, while containers are lifted by a crane on the installation. On average, approximately 75% of what is brought to the platforms is returned to the depot as backload in the form of bulk or containers on deck (Statoil, 2015c). The backload might be empty containers, used equipment going back to the depot or to external suppliers, drilling fluids, or basic items such as used towels.

Figure 2.4: The deck of an OSV. Personnel are getting cargo ready to be loaded onto the offshore installation. Photo: Nils B. Albjerk, February 2015.

Everything transported offshore, except bulk and pipes, is transported in containers of different sizes. This means that the number of containers going out also has to be brought back. Although outgoing supplies usually are heavier than the corresponding backload, baskets containing e.g. oil based drill cuttings might be heavier when returning to the depot because of higher concentration of liquids. At some installations, water based drill cuttings that can be dumped directly into the sea are used, giving no backload. All bulk products brought back to the depot

must be analyzed on the installation and approved by the captain on the vessel. This is done to prove the content and the origin of the load when the vessel arrives at the depot. If the mud has a high concentration of e.g. hydrogen sulfide, the tanks on the vessel might have to be cleaned before the next voyage. In a *voyage*, an OSV starts at the onshore supply depot, visits a set of offshore installations in a given sequence, and sails back to the depot.

When the backload is onboard, the vessels continue to other installations or the depot. When the vessels arrive at the depot, the backload is transferred onshore, and the vessels are prepared for their next voyage.

2.3 Planning processes in offshore logistics

In this section, the planning of routes and schedules for OSVs performed at the strategic, tactical, and operational level in Statoil is described.

At the strategic level, the properties of the OSV fleet are decided. Most OSVs are chartered on long- or medium-term contracts, that is, for more than 45 days. The fleet consisting of OSVs chartered for more than 45 days is called the long-term fleet in the rest of the report. OSVs in the long-term fleet are usually chartered at lower prices than the average spot market price. A vessel from the spot market is chartered for one to 45 days. As there are high costs associated with chartering OSVs, one of the most significant decisions to be made is what type, size, and number of OSVs the fleet should consist of. This problem is further discussed in Halvorsen-Weare, Fagerholt, Nonås, and Asbjørnslett (2012). In Statoil, deciding the optimal fleet is based on planning performed by optimization and supply chain management specialists. Historical data and expected future demand is used to estimate the necessary number of weekly visits to each offshore installation and necessary OSV deck capacity for each delivery. Optimal routes and schedules are then found by minimizing sailing times with respect to resource and capacity constraints. The necessary number of OSVs needed for the suggested routes and

schedules determines the fleet size. Additionally, the prices of OSVs from the spot market have to be compared to the costs of chartering the long-term fleet.

Since exact demand and delays due to weather conditions are impossible to predict, some measures to increase robustness are added to the mathematical models when these planning activities are performed. One of these measures is that the number of visits to installations on a voyage cannot exceed a given number. The probability of delays increases and the flexibility of being able to handle higher volumes than expected decreases as the number of visits per voyage increases.

As explained in Halvorsen-Weare et al. (2012), planning of routes for OSVs is done using a decision support tool based on mathematical models that determines the optimal weekly schedule for each vessel. The schedules are then continuously adapted at the onshore supply depots for every two weeks according to updated demand information. At the onshore supply depot, supplies are received from external distributors, prepared for shipment, sailing plans for each OSV are determined, loading of supplies onto OSVs, and unloading of backload from OSVs that return to the depot are conducted. Each depot serves several offshore installations, and allocates the available fleet according to the given routes and schedules. In most cases, routes and schedules need to be adapted due to unexpected demand, delays and priorities. *Priorities* are order requests from customers that may lead to significant costs if they are not serviced.

After the departure of an OSV from the depot, the responsibility for sailing plans and their necessary adjustments is given to an operational department at Statoil's offices in Bergen. This department is using equipment showing real-time positions of all OSVs, and communicates delays to OSVs and offshore installations, receives urgent orders, and finds new routes for OSVs according to priorities and delays. Currently, the adjustments made to sailing plans are based on the experience and judgment of professionals.

A summary of the planning activities with their purpose, methods, and output is shown in Table 2.1 and further explained in Chapter 4.

	Strategic planning	**Tactical planning**	**Operational planning**
Purpose	• Deciding optimal fleet composition. • Optimization of routes and schedules.	• Adapt routes and schedules to new demand information and delays. • Schedule loading and unloading of OSVs at onshore supply depot.	• Adapt routes and schedules to new demand information and delays. • Manage OSVs during sailing. • Manage loading and unloading operations at offshore installations.
Methods	• Optimization methods.	• Experience and judgement of employees at onshore supply depot.	• Experience and judgement of employees at operational central.
Output	• Optimal fleet composition. • 14 days plan based on expected demand.	• 14 days plan based on plan from strategic level, new demand, and delays.	• Actual routes to be sailed. • Actual pickups and deliveries to be made.

Table 2.1: Summary of the planning processes at strategic, tactical, and operational level. The focus of this report is on the operational level.

Chapter 3

Literature

This chapter contains a review of some of the literature relevant to the problem discussed and to the solution methods used in this report. Since the focus of the report is on operational planning and disruption management in the offshore supply service, some papers that consider disruption management in general routing problems are reviewed first in Section 3.1. Moving closer to the problem, some studies regarding pickup and delivery problems are reviewed in Section 3.2. Finally, some research on problems in the offshore supply service is presented in Section 3.3. By reviewing these problems, the most essential areas found in the literature relevant to the problem discussed in this report are covered.

Several articles and authors have been recommended to the authors by the supervisors for this report. In addition to these, Scopus searches has been used, and to some extent Google Scholar. Scopus (Scopus.com, 2015) is considered to be one of the best databases for academic papers.

3.1 Disruption management

Relatively few studies regarding disruption management in shipping are encountered, and to the authors' knowledge there exist no publications on disruption

management in the offshore supply service. However, thorough research is done on similar problems in airline and railway transportation. Although there are significant differences between shipping, airline transportation, and railway transportation, the disruption management problems contain to a large extent the same challenges and possible recovery actions. This section reviews disruption management studies from liner shipping, airline passenger transportation, and railway transportation relevant to the problem discussed in this report.

The vessel schedule recovery problem, discussed in Brouer, Dirksen, Pisinger, Plum, and Vaaben (2013), regards disruption management in container liner shipping. Different recovery actions are proposed, such as increasing speed, canceling deliveries, and swapping port visits. A model considering sailing costs, delays, and misconnecting cargo is presented, and it is run with data from real life cases. The results indicate that significant savings can be achieved when recovery actions are based on mathematical models instead of the experience and judgement of operations managers. In Brouer et al. (2013), only one other paper on disruption management in shipping is encountered. The authors of the paper claim that such research is often ordered by liner shipping companies, and that they may avoid publishing their results due to the competitiveness of the industry. In Kjeldsen (2012), a mathematical model for simultaneous rescheduling of ships and cargo in liner shipping is presented. The author mentions poor weather conditions, port congestion, low port productivity, towage, tidal windows, and several other sources of disruptions. Suggested recovery actions are among others changing the departure or arrival time at ports, transshipment of cargo between ports, and speed adjustments. The problem is modeled as a flow problem restricted by a time and space network, and is solved by a large neighborhood search heuristic.

The possible measures to handle disruptions in Brouer et al. (2013) are mostly similar to the ones available in disruption management in offshore supply. However, both Brouer et al. (2013) and Kjeldsen (2012) consider container liner shipping, in which there are usually great distances between the ports. The problem of

supplying offshore installations from an onshore supply depot involves very short distances compared to the problems discussed above. This implies that a measure such as increasing speed to reduce delays will be less effective than in liner shipping. The possibility of chartering additional OSVs from the spot market is an additional option available in disruption management in offshore supply. Also, the planning horizons of the liner shipping problems are longer than in the problem discussed in this report.

A significant amount of research has addressed disruption management in the airline industry, motivated by the major costs associated with delayed aircraft. Aircraft recovery problems are similar to disruption management problems in shipping in several ways. Problems in the airline industry consider aircraft and passengers, whereas shipping problems consider vessels and cargo, respectively. Both types of problems are extensions of traditional vehicle routing problems, such as the vehicle routing problem (VRP) and the vehicle routing problem with time windows (VRPTW). In Dienst, Røpke, and Vaaben (2012), two different models for the aircraft recovery problem is presented. The problem is represented by two multi-commodity network flow models. Airline schedule recovery models and algorithms considering aircraft, crew, and passenger recovery are proposed and demonstrated in Bratu and Barnhart (2006). The main objective in the paper is to find the optimal trade-off between airline operating costs and passenger delay costs. The presented decision model is run with simulated airline operations, and it is shown that by applying the model, significant reductions in passenger arrival delays can be achieved without increasing operating costs.

In aircraft recovery problems, there is often no delay at the beginning of each day, since few aircraft depart at night. The vessels discussed in this report are however operating 24 hours a day, several days in a row. Thus, it is harder for the vessels to get back on schedule compared to the aircraft.

In Huisman, Kroon, Lentink, and Vromans (2005), state-of-the-art operations research models and techniques used in passenger railway transportation are pre-

sented. As well as strategic, tactical, and operational planning, short-term planning where disruptions, such as speed reductions due to maintenance, are encountered are considered in the paper. This is similar to speed reductions due to harsh weather conditions in the offshore supply service. The authors argue that heuristics should be applied for real time control since decisions must be made within few minutes. A mixed integer formulation representing the problem of rescheduling trains in case of disrupted schedules is presented in Acuna-Agost, Michelon, Feillet, and Gueye (2011). Due to the difficulty of solving the problem exactly, the authors propose a limitation of the search space around the non-disrupted schedule by introducing local-branching-type cuts.

In railway transportation, emphasis is put on avoiding conflicting allocation of trains on railways and at stations. This may be compared to avoiding vessels arriving at the same time at offshore installations or at the onshore supply depot. Railway disruptions are similar to offshore supply disruptions in the sense that speed reductions might be encountered due to unforeseen events. However, delays on one train might propagate throughout the network and cause delays for other trains. The delay of a vessel can only disrupt other vessels' schedules when arriving at the same time at the depot or when conflicting arrivals at offshore installations occur.

3.2 Pickup and delivery problems

In the survey by Berbeglia, Cordeau, Gribkovskaia, and Laporte (2007), a classification scheme for static pickup and delivery problems are given. The classification is done by a three-field scheme; [structure|visits|vehicles]. The *structure* of the problems can be many-to-many (M-M), one-to-many-to-one (1-M-1), or one-to-one (1-1). (M-M) means that the vehicles transport goods from several customers to several customers. In (1-M-1)-problems, vehicles depart from a central depot to customers and finish its route back at the depot. Finally (1-1) means transporting

something from one customer to another. *Visits* describes the way the pickup and delivery operations are performed. One can either visit each customer once for simultaneous pickup and delivery (PD), visit the customers up to two times (P-D), or the customers might have either a pickup or a delivery (P/D). Finally, the last field, *vehicles*, gives the number of vehicles used to handle the transportation of goods in the problem. Whenever a field is undefined, "-" is used, and "m" is for problems with multiple vehicles. The problem studied in this report can be classified as 1-M-1|P-D|m according to the scheme, i.e., a pickup and delivery problem where all deliveries are transported from a central depot, all pickups are taken back to the central depot, and multiple vehicles are used. Offshore installations might be visited up to two times, either conducting pickup and delivery simultaneously or at different points in time.

In Dell'Amico, Righini, and Salani (2006), the vehicle routing problem with simultaneous distribution and collection is considered. It is described as reverse logistics, and relates to environmental concerns for recycling and reuse of goods and packages. Like the problem discussed in this report, the problem consists of supplying customers with goods from a central depot, while also collecting the waste goods and bringing it back to the depot. The problem can be classified as 1-M-1|PD|-. Customers are visited for simultaneous pickup and delivery, and the problem is different from the one considered in this report, since customers are not allowed to be visited twice. A branch-and-price solution method is proposed, and both exact dynamic programming and state space relaxation are used to solve the pricing subproblem. Suitable branching strategies are chosen for the exact solution method and for the state space relaxation method. The state space works as a projection by reducing the dimension of the problem. The paper concludes that branch-and-price is a viable approach for problem instances of small and medium size.

In Brønmo, Christiansen, Fagerholt, and Nygreen (2007), a problem that maximizes profits for a pickup and delivery problem of bulk cargoes in tramp shipping is

considered. A set partitioning approach and a multi-start local search are presented and compared. The problem can be classified as M-M|P/D|m, i.e a many-to-many pickup and delivery problem where each node is either a pickup or a delivery node. In Korsvik, Fagerholt, and Laporte (2009), the same planning problem as Brønmo et al. (2007) is solved using a tabu search heuristic. In the search for solutions, they make it possible to explore infeasible regions of the solution space. The best initial solutions are used in the tabu search iterations and in the intensification phase. The tabu search heuristic gives better solutions compared to the multi-start local search proposed in Brønmo et al. (2007). The problem varies from the one considered in this report in different ways. It is a many-to-many problem and since a node is either pickup or delivery, there is no possibility of a node being visited twice or for simultaneous pickup and delivery.

In Gribkovskaia, Halskau, Laporte, and Vlcek (2007), a mathematical model for the single vehicle routing problem with pickup and deliveries is introduced. In this problem, the customers can either be visited twice where pickup and delivery are performed separately or once where these two operations are combined, and the vehicle must start at and return to a central depot. Thus, it is classified as a 1-M-1|P-D|1-problem. It is mentioned that the linear relaxation yields a poor lower bound and only small instances can be solved to optimality. The model can however be used to analyze properties and shapes of the optimal solutions. Also, the concepts of general solutions such as hamiltonian paths, double-paths, and lasso paths are discussed. Since only small instances can be solved by the mathematical model, heuristics are introduced. The solution concepts introduced in the paper are feasible solutions for the model presented in this report as well. A central depot is also used as in the problem considered in this report, and they come to the same conclusion as many other papers have; that only small instances can be solved without the use of heuristics.

The recent study presented in Polat, Kalayci, Kulak, and Günther (2015) considers the VRP with simultaneous pickup and delivery (VRPSPD) with time limits.

Using the classification by Berbeglia et al. (2007) this problem can be described as 1-M-1|PD|m. The problem described in this report is quite similar to the one in Polat et al. (2015) except that in the problem considered in this report, it is possible to visit customers twice. In the problem in Polat et al. (2015) however, each of the customers has to be served once for both pickup and delivery by a given fleet of identical vehicles. The vehicles leave the central depot carrying the total amount they have to deliver, returning with the total amount that was picked up. Total voyage time is the sum of total travel time and service time at customers. Furthermore, the voyage must be finished before the maximum allowed voyage time, which also is the case for the problem in this report. An initial solution is constructed by use of the Clarke Wright savings algorithm, followed by a variable neighborhood search (VNS) to improve the initial solution over several iterations. Finally, a perturbation mechanism is used to escape local optima. The neighborhoods are created using four inter-route and four intra-route operators.

The problem discussed in this report is a simultaneous pickup and delivery problem, and can be related to the work mentioned above. As problem instances increase in size, it might be difficult to solve large instances of the problem within reasonable time using an exact method. On an operational planning level, solution time is an important aspect, and solving the problem using a heuristic might be the only viable option for all practical purposes.

Using the classification presented in Berbeglia et al. (2007), several of the papers mentioned above have the same structure and/or visiting sequence as the problem considered in this report. In Dell'Amico et al. (2006), a similar problem in reverse logistics is studied, but the authors conclude, as in many other papers, that an exact solution is only viable for small to medium sized instances. This motivates the use of heuristics for solving larger problem instances in this report. The most similar problem to the one discussed in this report is found in Polat et al. (2015), since it is a simultaneous pickup and delivery problem with time limits on the vehicles. The problems differ due to the number of times each customer might be

visited and the possibility of chartering extra vessels. The recent publication, along with the strong results provided by the proposed heuristic, makes it very relevant for the work presented in this report.

3.3 Offshore supply

Several papers address ship routing and logistics in the upstream supply chain for petroleum production. In Halvorsen-Weare et al. (2012), a strategic problem is discussed, which consists of finding the optimal fleet composition at an onshore supply depot for servicing a set of offshore installations, and of finding optimal routes and schedules for OSVs. As opposed to the problem discussed in this report, only outgoing supplies are considered and not cargo returning to the onshore supply depot. The authors conclude that a voyage-based solution approach can be used on real life problems faced by the oil and gas company Statoil.

In Fagerholt and Halvorsen-Weare (2011), the same problem as in Halvorsen-Weare et al. (2012) is considered. The paper presents two different mathematical models, first an arc-flow model and then a voyage-based approach of describing the problem. Time windows representing opening hours at installations are included in the models. Both model formulations are applied to several real life cases for Statoil, including up to 14 offshore installations. In both Fagerholt and Halvorsen-Weare (2011) and Halvorsen-Weare et al. (2012), a ship routing problem similar to the one discussed in this report is considered. However, both papers regard strategic decisions, while this report considers the ship routing problem from an operational point of view. In the strategic problems, the OSV fleet composition is to be decided, while it is given in the problem discussed in this report.

In Shyshou, Gribkovskaia, Laporte, and Fagerholt (2012), the same problem as in Fagerholt and Halvorsen-Weare (2011) is studied. They present a large neighborhood search heuristic. The large neighborhood search consists of rearranging a large portion of the solution and hence exploring greater regions of the solutions

space. The heuristic uses many procedures to improve the solutions, e.g. removing and inserting nodes in a route, reducing the number of voyages and vessels, reducing voyage durations, and performing intra-voyage optimization.

Also in Maisiuk and Gribkovskaia (2014), the problem of deciding the optimal fleet composition in offshore supply service is addressed. The paper presents a discrete-event simulation model, where uncertainty in sailing and service duration is considered. The study verifies an oil company's decision of having four OSVs on long-term contracts.

In Sopot and Gribkovskaia (2014), a pickup and delivery problem describing logistics in the upstream supply chain for offshore oil and gas production is considered. A mathematical model for a single vehicle ship routing problem with multiple commodities is presented, where all offshore installations are serviced from a single depot. Thus, the problem can be classified as 1-M-1|P-D|1, as explained in Section 3.2. The problem is similar to the one considered in this report in its structure, but it only considers one vessel and allows multiple visits since the solutions is a non-hamiltonian route. The problem is solved using a metaheuristic, and it is shown that the algorithm outperforms a commercial optimization solver in speed and produces solutions of high quality.

In Gribkovskaia, Laporte, and Shlopak (2007) a pickup and delivery problem consisting of servicing offshore oil and gas platforms in the Norwegian Sea is presented, where a single vessel is to conduct the entire voyage. The solution is obtained by the use of a tabu search heuristic, since exact solution methods for pickup and delivery problems are usually limited to small problem instances. Both this and the problem mentioned above are described as one-to-many-to-one problems where certain customers can be visited at most once, while others can be visited at most twice on a voyage. Thus, the appropriate classification for the two problems is 1-M-1|P-D|1.

Most of the existing research considering logistics in the upstream supply chain for offshore oil and gas production regard strategic decisions, while this report

discusses operational planning and challenges related to this. Significant costs are related to the use of OSVs from the spot market and to disrupted routes and schedules. These costs may be reduced if operational decisions are supported by decision tools based on optimization methods. As the authors of several papers conclude, the pickup and delivery problem is difficult to solve for large problem instances. Thus, a solution approach applying heuristics may be an appropriate measure towards increasing the solution speed.

Chapter 4

Problem description

This chapter contains a detailed description of the problem studied in this report. A short problem summary is given at the end of the chapter.

The problem studied in this report consists of deciding new routes and schedules for OSVs, and how disruptions to planned routes and schedules can be handled. Disruptions can be caused by adverse weather conditions, unexpected orders from offshore installations, and uncertainty in the volume of orders. As a result of disruptions, OSVs do often not sail and operate according to the planned routes and schedules. A decision tool based on optimization methods to support the operation of vessels may help to reduce the resulting negative effects. It may also simplify the planning process and lead to faster decision making.

Planned departure time from the onshore supply depot and planned duration of voyages are given as input to the problem. As mentioned in Section 2.2, a voyage for an OSV starts at the onshore supply depot, visits a set of offshore installations in a given sequence, and sails back to the depot. An example with two OSVs, one voyage for each vessel, and four offshore installations is shown in Figure 4.1.

Chartering and operating OSVs cause major costs in the upstream supply chain of offshore oil production. Since Statoil usually does not own any OSVs, the vessels

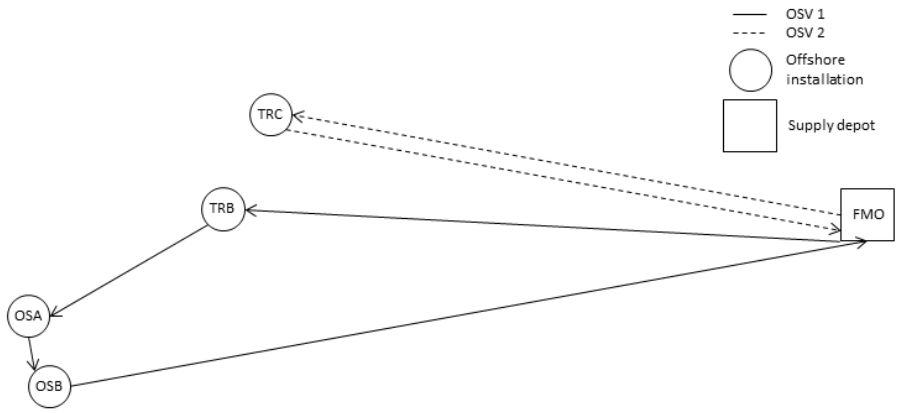

Figure 4.1: Example voyages for two OSVs and four offshore installations.

are chartered from ship owning companies. Most OSVs supplying offshore installations are chartered on contracts for more than 45 days. Due to uncertainty in demand and delays in ongoing operations, OSVs are also chartered from the spot market on short-term, which is usually more expensive. According to employees in Statoil, daily chartering costs for OSVs on long- or medium-term contracts typically lie between NOK 110 000 and NOK 140 000, while daily costs for chartering vessels from the spot market typically lie between NOK 150 000 and NOK 400 000. If the long-term fleet can be used more efficiently, and the use of spot market OSVs can be reduced, significant cost reductions in the upstream supply chain may be achieved.

It is preferable to utilize the capacity of each OSV to as large extent as possible in order to minimize the number of vessels needed to serve the offshore installations. Therefore, planning routes and schedules is done with emphasis on minimizing total costs for fuel consumption and chartering of OSVs from the spot market. If an OSV arrives at the onshore supply depot after the planned schedule due to disruptions, it may affect the departure time of the next planned voyage. In addition, orders that are postponed due to disruptions to the planned routes and schedules can affect operations at the offshore installations.

Delayed supplies can be critical in the offshore production of oil and gas since it in a worst-case scenario may lead to a halt in production. This is in general far more expensive than chartering OSVs from the spot market. The costs of halt in production is estimated to be around NOK 200 000 per hour, and if Statoil does not own the rig, rental costs for an idle rig is added to this figure as well. There may also be repercussions further down the supply chain. For example, if the refinery at Tjeldbergodden is not supplied with gas from the Heidrun field, they have to restart their production processes, which can take several days.

The offshore installations are either open for unloading and loading of cargo 24 hours a day or between 0700 and 1900. If an OSV arrives at an offshore installation when it is closed for loading operations, the vessel will have to wait until the installation opens the next day or sail to another installation. However, if the loading operation has started within reasonable time before closing, some overtime on the installation is accepted to complete the operation. The onshore supply depot is open for loading operations 24 hours a day in order to serve all associated offshore installations, but it is still only able to serve a limited number of vessels per day.

Planned routes and schedules are based on estimated demand from each offshore installation. An example illustrating a case with seven offshore installations and three OSVs is shown i Figure 4.2. Both bulk cargo and cargo on deck demand capacity from the OSVs. However, the installations are often flexible regarding the amount of bulk products they receive. Thus, the amount of bulk products delivered to each installation can be adjusted to fit the capacity of the OSVs. On the contrary, the size of the containers on deck to be delivered cannot be adjusted. Therefore, the OSV deck capacity is considered to be more limiting for routes and schedules than the bulk capacity.

The deck capacity is reduced due to zones that must be free of cargo to allow space for tubes to unload bulk, and there must be space between the cargoes to allow the deckhands to move freely on the deck.

Historical data show that deck capacity is usually not the binding resource constraint when plans are developed at strategic level in Statoil. The plans are rather limited by the maximum number of visits an OSV can make during a voyage. A maximum number of visits is introduced in order to reduce the probability for deviations from the planned routes and schedules, caused by adverse weather conditions, unexpected orders, and uncertainty in order volume. However, when considering operational challenges, deck capacity is considered to be more important than the maximum number of visits per voyage. Deck capacity is especially an issue after a period of extreme weather, where the OSVs have been unable to sail, and have to make up for unserved orders by utilizing their deck capacity to as large extent as possible. In addition, more information is available in operational planning than in strategic planning. Therefore, when routes are actually sailed, the maximum number of visits may be ignored in order to serve as many orders as possible, given new information about unexpected orders, weather conditions and order volume.

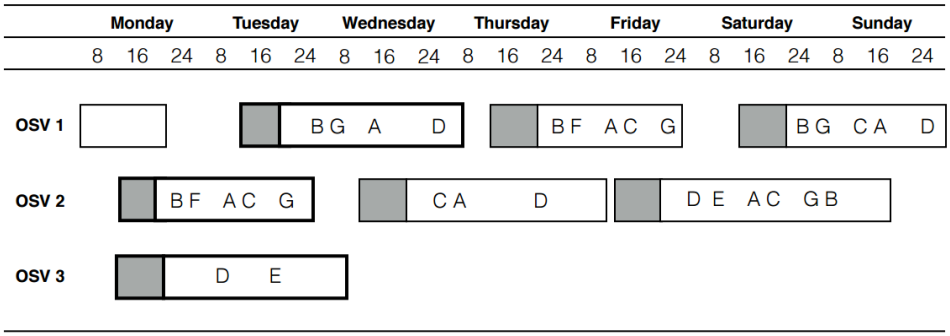

Figure 4.2: Possible weekly schedule for three OSVs and seven offshore installations (A-G). The dark squares illustrate the time needed for loading at the supply depot. A set of voyages that could be considered in this problem is marked by thick lines.

Planning routes and schedules is complicated by uncertain parameters such as demand and parameters affected by the weather conditions. Adverse weather

conditions cause major challenges in the process of supplying offshore installations, especially during the winter season. Sailing speed of the OSVs is reduced when waves are higher than 3.5 meters according to Fagerholt and Halvorsen-Weare (2011). Under extreme weather conditions the vessels may not be able to sail at all. Loading and unloading operations at the offshore installations are even more sensitive to weather than the sailing speed. Loading and unloading operations at offshore installations are only possible until the waves reach a height of around 4-5 meters, or the winds exceed 40 knots, due to limitations on the cranes on the installations.

The uncertainty of demand depends on what kind of operations the offshore installations perform. While the supplies needed for regular extraction of oil and gas is relatively predictable, drilling does more often cause urgent and unexpected orders from platforms. The volume of the demand is also hard to predict, leading to unused capacity or rerouting because of full vessels and to OSVs being needed from the spot market. In case of urgent orders, even helicopters may be used to supply the offshore installations. Furthermore, transshipment orders can occur during the voyage of OSVs, meaning rerouting of the vessel to pick up goods at one installation and deliver it at another installation.

The demand volume has an impact on the service time for the OSVs. It is estimated that it is possible to do ten lifts from an OSV to an installation and ten lifts from an installation to an OSV with a crane per hour. The time needed for transferring bulk products vary, since heavier products, such as barite, take more time to send through the tubes than for instance clean water. Different kinds of bulk are stored in different tanks on board the OSV, and different tubes must be used to transfer them between the vessel and the installation. This might require the OSV to change their position at the installation, using more time. It is also required that all bulk that is delivered from the installation to the OSV is analyzed, documented, and accepted before loading can begin. If this is not done, the OSV will not be able to unload the bulk when returning to the depot.

Changing the crew on an offshore installation is done solely by helicopters and is a costly operation. If an OSV is conducting loading operations while there is a helicopter approaching, the OSV's operation is usually cancelled due to the fact that the helicopter pilots do not want vessels in the flight approach area. This might result in rerouting of the OSV for the duration of the helicopter visit. The exception is if the OSV has started transferring bulk products to or from the installation, then the loading operation continues.

When delays, unexpected orders, or too high volume of orders occur, there are a few different options for how to handle them. The goal is to choose the option that minimizes total costs of OSV chartering, fuel consumption, and costs of negative effects for production at the offshore installations. The different options are as follows:

- **Omit deliveries:** Omitting visits to certain offshore installations may get the OSV back on schedule for the remaining route in case of delays. However, worst-case scenario when choosing this option is halt in production at an offshore installations because of missing supplies. It is therefore not preferable to omit any deliveries, unless the supplies have low priority and the consequences are insignificant for production.

- **Speed changes:** Speeding up OSVs may compensate for delays on a route, but the applicability of this option is limited since weather conditions may make it impossible to increase speed. If an increase of sailing speed is possible, there is still no guarantee that the OSV will reach the next offshore installation in time, since the sailing distances are relatively short.

- **Swap order of visits:** Swapping the order in which offshore installations are visited may be a desirable option in case one delivery is more urgent than another. On the other hand, the offshore installations prefer to know the time of deliveries in order to plan their own operations.

- **Utilize slack:** The given schedules for the OSVs contain some slack to compensate for uncertainty in the duration of sailing and loading operations. The slack may be utilized if some delay on the remaining voyage can be allowed. If there is available capacity on the OSV and it is still at the onshore supply depot, this option may also be chosen in case of unexpected orders.

- **Utilize free fleet capacity:** In case of a delayed OSV and available capacity on another OSV at the onshore supply depot, the latter may be used to supply one or more of the offshore installations on the next voyage of the delayed vessel. Utilization of free capacity should also be done in case of unexpected orders or if the total volume of the orders is too big for one OSV to handle.

- **Charter OSV from spot market:** If there are no available OSVs in the current fleet, additional ones may be chartered from the spot market. This may be costly depending on the spot market price.

The planning horizon for the problem is defined by the duration of the voyages sailed by the OSVs. Statoil has decided that planned duration of a voyage should not be longer than two days if the voyage starts at Monday, Tuesday, Wednesday or Thursday. If the voyage starts on either Friday or Saturday, the duration may be three days or less. No voyages start on Sundays. When solving the problem, at most one voyage for each OSV is considered.

Problem summary

The problem in this thesis considers deciding new routes and schedules in an operational planning setting, and how to handle disruptions to current routes and schedules for OSVs. Changing planned routes and schedules can be done by utilizing the options listed above, or a combination of these. The planned operations at offshore installations may be affected when the service of OSVs deviate from the planned schedule. Simultaneously, significant reductions in sailing costs can

be made by finding new routes for the OSVs, given disrupted sailing plans and schedules. When solving the problem, the disadvantage of postponing the service to offshore installations, delaying vessels, and costs related to sailing, loading operations, and chartering OSVs from the spot market should be balanced. Thus, the first objective of the problem is to minimize sailing and service costs, and costs of chartering OSVs from the spot market. The second objective is to minimize the time OSVs arrive at the onshore supply depot after schedule. Finally, the third objective is to maintain a sufficient service level for the offshore installations by servicing as many of the planned orders as possible. The multi-objective problem is constrained by the deck capacity of OSVs, the available time for OSVs, and the number of available OSVs in the fleet.

Chapter 5

Mathematical formulation

In this chapter, the mathematical formulation of the problem considered in this report is presented. The assumptions which the mathematical model is based on are listed in Section 5.1. Then, in Section 5.2 the notation is introduced, and finally, an arc-flow formulation is presented in Section 5.3.

5.1 Modeling assumptions

In the following, the assumptions made when developing the mathematical models are listed. Some assumptions are made to reduce the computational complexity of the problem, and some are made because of incomplete or uncertain information. The assumptions are chosen with emphasis on not affecting the applicability of the solution.

- The problem is modeled as a pickup and delivery problem. The mathematical models presented in Fagerholt and Halvorsen-Weare (2011) assume that there will be enough capacity to transport backload from the offshore installations to the onshore supply depot. Since the problem discussed in this report is seen from an operational perspective, and backload is about 75% of the

outgoing supplies on average (Statoil, 2014a), the mathematical models will take backload into account. The volume of backload varies, and may in some cases be higher than the outgoing supplies.

- Each delivery and each pickup order will only be served by one vessel. In theory, demanded cargo that is brought to offshore installations could be split between several vessels. E.g., if one delivery order consists of two containers, these could be transported by two different vessels. However, this is in most real life cases not practical, and therefore it is reasonable to assume that each order is served by one vessel only.

- It is assumed that each offshore installation will have at most one pickup and one delivery request. If an installation places several pickup or delivery orders, these can be combined into one order to fit the model, provided that the combined order's size does not exceed the vessel capacity.

- When an unexpected order is reported by an offshore installation, one pickup and/or one delivery node can be added to the problem instance. If the offshore installation has already placed an order, it can be sufficient to add the volume of the new order to the first order. That is, if the OSV that is supposed to service the first order has not left the onshore supply depot. If the OSV has left the depot, one pickup and/or one delivery node should be added.

- In practice, the OSVs carry both cargo on deck and bulk cargo below deck. According to OSV personnel, deck capacity is usually the binding capacity resource. Therefore, bulk cargo is not considered in the mathematical model.

- Some of the offshore installations considered in the problem are only open for loading operations during day time. To include this in the mathematical models, time windows could have been introduced. Some Statoil representatives argue that the installations should be kept open at night, since this is less expensive than having to adapt the schedules to the opening hours

or having OSVs waiting for them to open. Thus, since all installations may be open 24 hours a day in the future, time windows are not included when modeling the problem in this report.

- The cost representing a loading operation is only represented by the fuel consumption while an OSV is waiting by an offshore installation. In real life, there are also costs related to lifting cargo between the vessel and the installation, and to staff involved in the loading operations. These costs are not significant and may be considered as fixed costs, therefore they are not included in the model.

- The speed of the OSVs is assumed to be constant. In practice, the speed is highly influenced by weather conditions, and the crew at each OSV is able to adjust the speed to some extent if their vessel deviates from schedule. Since the sailing distances for the OSVs are relatively small, this assumption is not likely to affect the order in which installations are visited, and is therefore not significant to the solution of the problem. To compensate for this assumption, the estimated speed and travel times for each vessel can be changed due to the current weather conditions when the model is solved on real life cases.

- In case the available fleet of OSVs is not sufficient to serve all orders, it is possible to charter additional vessels from the spot market. If this is needed, it is assumed that one OSV from the spot market will cover the remaining orders that the fleet cannot serve. This assumption can easily be avoided by adding additional variables to the mathematical model and to the set of available OSVs from the spot market. More than one OSV from the spot market will not be needed in most real life cases due to the short planning horizon of the considered problem.

- The costs associated with chartering the long-term fleet is considered as sunk when developing the mathematical model, since this report considers an op-

erational problem in which the long-term fleet cannot be changed.

- The costs associated with postponing orders from offshore installations and with OSVs arriving at the onshore supply depot after schedule are represented in the mathematical model by adding penalty costs to the objective function. The penalty cost parameters are estimates based on parameter testing and advice from Statoil employees.

- In the similar problem discussed in Fagerholt and Halvorsen-Weare (2011) and Halvorsen-Weare et al. (2012), the number of possible visits made by an OSV during a voyage is restricted by an upper limit. In strategic planning, the upper limit is introduced in order to increase the robustness of routes and schedules, since a high number of visits during a voyage gives high uncertainty in sailing times. In operational planning, the planned routes and schedules can be less robust, since more information is available. Therefore, it is not considered as reasonable to include the upper limit on the number of visits in a voyage in the mathematical model.

- In order to use the mathematical formulation for voyages that have already started, the variables representing the already sailed distances can be fixed.

5.2 Notation

The onshore supply depot and the offshore installations are represented by a network of nodes. The set $\mathcal{N} = \{0, ..., n+m\}$ consists of all nodes in the network. The set $\mathcal{N}^P = \{0, ..., n\}$ contains all pickup nodes, where node 0 represents the depot pickup node. The set $\mathcal{N}^D = \{n+1, ..., n+m\}$ contains all delivery nodes, where node $n+1$ represents the depot delivery node. An offshore installation will have an associated pickup node if transport of backload to the depot is demanded. Similar, it will have an associated delivery node if delivery of supplies to the installation is demanded. If offshore installations place urgent orders and the associated pickup

and/or delivery nodes do not already exist in the network, the additional nodes are added. If urgent orders are placed and the associated nodes already exist, the urgent orders are added to the already existing orders. The set $\mathcal{V} = \{1, ..., k+1\}$ contains all available OSVs, where the OSV $k+1$ represents the one OSV from the spot market.

The parameter C^S_{vij} represents the costs associated with sailing from node i to node j and servicing node j for OSV v. When an OSV is just chartered or is to be delivered back to the company owning the vessel, additional preparation and cleaning is needed. The associated cost can be added to $C^S_{(k+1)0i}$ for all $i \in \mathcal{N} \setminus \{n+1\}$. OSVs from the spot market are chartered for a number of whole days, and the daily time charter rate is represented by C^{TC}. The penalty cost C^R_i is associated with postponing the service to a node i. The cost induced by a postponed pickup or delivery order depends on the importance and urgency of each order. The penalty cost per hour when an OSV v arrives at the depot after schedule is denoted by C^D_v.

The size of an order for a delivery node i is denoted by the parameter D_i. The size of an order for a pickup node i is denoted by the parameter P_i. The size of orders is measured in square meters, and Q_v is the total deck capacity of OSV v measured in square meters.

The time (hours) needed for sailing from node i to node j and servicing node j for OSV v is denoted by T_{vij}. For all parameters $T_{(k+1)0i}$, that is, the time needed for the spot vessel $k+1$ to sail from the supply depot to any installation i, additional time can be added for preparation and cleaning of the vessel.

The next possible departure time from the depot for OSV v is T^{MIN}_v, and T^{MAX}_v is the planned arrival time at the depot for the next voyage for OSV v. The OSV $k+1$ must be chartered for a whole number of time periods (days), where the length of a time period is denoted by the parameter H. The parameter τ denotes the maximum allowed delay for the OSVs in the long-term fleet.

5.3 Arc-flow model

This section presents the arc-flow formulation, with variables, objective, and constraints, and a short model discussion.

The variable x_{vij} equals 1 if OSV v sails from node i to node j, and 0 otherwise. The support variable y_{vi} equals 1 if OSV v visits node i, and 0 otherwise. If the order at node i is postponed, the variable u_i equals 1, and 0 otherwise. The cargo load variables l_{vij} equals the load measured in square meters on OSV v when sailing from node i to node j, and if the arc is not sailed, it equals 0. The number of hours OSV v arrives at the depot after schedule on the next voyage is represented by t_v^D. The number of whole days that the OSV $k+1$ from the spot market needs to be chartered is denoted by t^{TC}.

Objective

$$min \sum_{v \in \mathcal{V}} \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} C_{vij}^S x_{vij} + C^{TC} t^{TC} + \sum_{i \in \mathcal{N}} C_i^R u_i + \sum_{v \in \mathcal{V} \setminus \{k+1\}} C_v^D t_v^D \qquad (5.1)$$

The objective function (5.1) consists of four parts. The first part summarizes the costs related to sailing and servicing nodes for all OSVs and for all arcs in the considered voyages. The second part express the variable cost related to chartering an OSV from the spot market. The third part summarizes the costs associated with orders that are postponed until a later voyage, and is included to maintain a sufficient service level at the offshore installations. The fourth part summarizes the costs related to OSVs in the long-term fleet that return to the onshore supply depot after planned schedule, and is included to reduce the delay propagating through consecutive voyages.

Sailing and service constraints

$$\sum_{i \in \mathcal{N} \setminus \{0\}} x_{v0i} = 1 \qquad v \in \mathcal{V} \qquad (5.2)$$

$$\sum_{i \in \mathcal{N} \setminus \{n+1\}} x_{vi(n+1)} = 1 \qquad v \in \mathcal{V} \tag{5.3}$$

$$\sum_{i \in \mathcal{N} \setminus \{0\}} x_{vi0} = 0 \qquad v \in \mathcal{V} \tag{5.4}$$

$$\sum_{j \in \mathcal{N}} x_{vji} - \sum_{j \in \mathcal{N}} x_{vij} = 0 \qquad v \in \mathcal{V}, i \in \mathcal{N} \setminus \{0, n+1\} \tag{5.5}$$

$$y_{vi} - \sum_{j \in \mathcal{N} \setminus \{i\}} x_{vij} = 0 \qquad v \in \mathcal{V}, i \in \mathcal{N} \tag{5.6}$$

$$\sum_{v \in \mathcal{V}} y_{vi} + u_i = 1 \qquad i \in \mathcal{N} \setminus \{0, n+1\} \tag{5.7}$$

Constraints (5.2) and (5.3) ensure that all voyages begin and end at the depot, respectively. Constraints (5.4) ensure that no arcs end at the depot pickup node. Constraints (5.5) are the sailing flow conservation constraints, and the support variables are set by (5.6). Constraints (5.7) ensure that all nodes are either serviced by an OSV or the order is postponed until a later voyage.

Capacity and cargo flow constraints

$$l_{vij} \leq (Q_v - P_j) x_{vij} \qquad v \in \mathcal{V}, i \in \mathcal{N}, j \in \mathcal{N}^P \tag{5.8}$$

$$l_{vij} \leq Q_v x_{vij} \qquad v \in \mathcal{V}, i \in \mathcal{N}, j \in \mathcal{N}^D \tag{5.9}$$

$$l_{vij} \geq P_i x_{vij} \qquad v \in \mathcal{V}, i \in \mathcal{N}^P, j \in \mathcal{N} \tag{5.10}$$

$$l_{vij} \geq D_j x_{vij} \qquad v \in \mathcal{V}, i \in \mathcal{N}, j \in \mathcal{N}^D \tag{5.11}$$

$$l_{vij} \geq (P_i + D_j) x_{vij} \qquad v \in \mathcal{V}, i \in \mathcal{N}^P, j \in \mathcal{N}^D \tag{5.12}$$

$$\sum_{i \in \mathcal{N}} l_{vij} + P_j x_{vjh} - l_{vjh} + Q_v x_{vjh} \leq Q_v \qquad v \in \mathcal{V}, j \in \mathcal{N}^P, h \in \mathcal{N} \tag{5.13}$$

$$\sum_{i \in \mathcal{N}} l_{vij} - D_j x_{vjh} - l_{vjh} + Q_v x_{vjh} \leq Q_v \qquad v \in \mathcal{V}, j \in \mathcal{N}^D, h \in \mathcal{N} \tag{5.14}$$

$$\sum_{j \in \mathcal{N}^D} D_j y_{vj} - l_{v0i} + Q_v x_{v0i} \leq Q_v \qquad v \in \mathcal{V}, i \in \mathcal{N} \tag{5.15}$$

$$l_{vi(n+1)} - \sum_{j \in \mathcal{N}^P} P_j y_{vj} + Q_v x_{vi(n+1)} \leq Q_v \qquad v \in \mathcal{V}, i \in \mathcal{N} \tag{5.16}$$

Constraints (5.8) and (5.9) ensure that if an arc is sailed by an OSV, then the load on board should not exceed the capacity of the vessel. Constraints (5.10) - (5.12) give a lower bound on the load on board depending on whether an OSV sails from a pickup node, to a delivery node, or both. Constraints (5.13) and (5.14) are the cargo flow conservation constraints. Since the model does not distinguish between cargo that is to be delivered to an installation and backload, constraints (5.15) ensure that the total amount of cargo to be delivered to installations on a voyage equals the load on board when the OSV leaves the depot. Similarly, constraints (5.16), together with (5.8) and (5.9), ensure that the load on board, when the OSV arrives at the depot, equals the total amount of picked up cargo on a voyage.

Spot market OSV constraint

$$t^{TC} \geq \left(\sum_{i \in N} \sum_{j \in N} T_{(k+1)ij} x_{(k+1)ij} \right) \frac{1}{H} \tag{5.17}$$

If the spot market OSV is needed, constraint (5.17) calculates the time it is used, and rounds up to the nearest whole day.

Time constraints

$$T_v^{MIN} + \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} T_{vij} x_{vij} - T_v^{MAX} \leq t_v^D \qquad v \in \mathcal{V} \setminus \{k+1\} \tag{5.18}$$

$$t_v^D \leq \tau \qquad v \in \mathcal{V} \setminus \{k+1\} \tag{5.19}$$

Constraints (5.18) set the delay variable, and constraints (5.19) assure that the delay is no more than τ hours for each OSV in the long-term fleet. The model in this report does not allow spot vessels to be delayed.

Subtour eliminating constraints

$$\sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S}} x_{vij} \leq |\mathcal{S}| - 1 \qquad v \in \mathcal{V}, \mathcal{S} \subset \mathcal{N}, |\mathcal{S}| \geq 2 \tag{5.20}$$

Constraints (5.20) are the subtour eliminating constraints.

Binary, non-negativity and integer constraints

$$x_{vij} \in \{0, 1\} \qquad v \in \mathcal{V}, i \in \mathcal{N}, j \in \mathcal{N} \tag{5.21}$$

$$y_{vi} \in \{0, 1\} \qquad v \in \mathcal{V}, i \in \mathcal{N} \tag{5.22}$$

$$u_i \in \{0, 1\} \qquad i \in \mathcal{N} \tag{5.23}$$

$$l_{vij} \geq 0 \qquad v \in \mathcal{V}, i \in \mathcal{N}, j \in \mathcal{N} \tag{5.24}$$

$$t_v^D \geq 0 \qquad v \in \mathcal{V} \setminus \{k+1\} \tag{5.25}$$

$$t^{TC} \in \mathbb{Z}^+ \tag{5.26}$$

Constraints (5.21) - (5.26) are the binary, non-negativity, and integer constraints.

The load on deck of an OSV v sailing from node i to node j is denoted l_{vij}. In order to reduce the number of variables, the load on board an OSV v when leaving a node i could instead be expressed as l_{vi}. However, the first formulation is chosen, since it may give a tighter formulation of the problem.

If an OSV v arrives before or after schedule at the onshore supply depot, the parameter T_v^{MIN} should be adjusted accordingly before the problem containing the next voyage for OSV v is solved.

Consider a voyage sailed by OSV v containing the two nodes $p \in \mathcal{N}^P$ and $d \in \mathcal{N}^D$ belonging to the same offshore installation. Assuming that the maximum load carried by v along the voyage is $l_{i^*j^*}$ on arc (i^*, j^*). Then,

$$l_{i^*j^*} = \sum_{i \in \Omega^P} P_i + \sum_{i \in \Theta^D} D_i,$$

where the set $\Omega^P \subseteq \mathcal{N}^P$ is the set of pickup nodes visited before j^*, and $\Theta^D \subseteq \mathcal{N}^D$ is the set of delivery nodes visited after i^*. The positions of i^* and j^* are denoted by I and J, respectively. N and M indicate the positions of p and d along the voyage, respectively, with $N < M$. Let l^1 denote the component of $l_{i^*j^*}$ generated by the nodes p and d. Let l^2 denote the component of $l_{i^*j^*}$ generated by p and d if p is placed at position M and d is placed at position N, while the positions of all the other nodes in the voyage remain unchanged.

Proposition 5.1. *$l^2 \leq l^1$, that is, visiting node p before node d never gives a better solution than visiting d before p.*

Proof.

$$J \leq N \implies p \notin \Omega^P \wedge d \in \Theta^D \implies l^1 = l^2 = D_d \tag{5.27}$$

$$I \geq N \wedge J \leq M \implies p \in \Omega^P \wedge d \in \Theta^D$$

$$\implies l^1 = P_p + D_d \wedge l^2 = 0 \tag{5.28}$$

$$I \geq M \implies p \in \Omega^P \wedge d \notin \Theta^D \implies l^1 = l^2 = P_p \tag{5.29}$$

The values of l^1 and l^2 only differ in (5.28), where $l^2 < l^1$. Thus, Proposition 5.1 is correct. □

All arcs going from the pickup node to the delivery node for the same offshore installation are removed when implementing the model. This reduces the number of variables and the number of possible voyages without reducing the possibility of finding an optimal solution.

Chapter 6

Voyage-based solution method

The arc-flow formulation of the problem presented in the previous chapter is difficult to solve for problem instances of realistic size using commercial optimization software. In order to solve real life problem instances, other solution approaches must be applied. This chapter presents a solution method based on pregeneration of voyages through dynamic programming and a voyage-based formulation of the problem. The proposed solution method aims at reducing the computational time needed to find optimal solutions. The voyage-based formulation is presented in Section 6.1, followed by a description of the dynamic programming approach applied for pregeneration of voyages in Section 6.2.

A similar voyage-based approach is presented in Fagerholt and Halvorsen-Weare (2011) and Halvorsen-Weare et al. (2012), where a strategic offshore supply problem of finding optimal fleet composition is considered. In Halvorsen-Weare et al. (2012), a full enumeration procedure for voyage generation is described. The approach presented in this report finds all possible voyages for all OSVs, and removes all infeasible and dominated voyages. In Halvorsen-Weare et al. (2012), only the duration is considered when finding the optimal path among a subset of nodes, while both duration and costs are considered in the dynamic programming ap-

proach presented here.

A schematic overview of the solution method is shown in Figure 6.1. Input data based on real life problem instances are used to generate all possible voyages that are feasible with respect to the available time and the deck capacity for each vessel. The voyages are generated using dynamic programming. When all possible voyages are found, the problem is solved using a voyage-based formulation of the problem. When input parameters change, the input data should be updated, new voyages should be generated, and the voyage-based model should be run again. When changing the penalty costs for delayed OSVs and postponed orders, only the input data for the voyage-based model needs to be updated, and the voyage-based model can be run again with the same generated set of possible voyages.

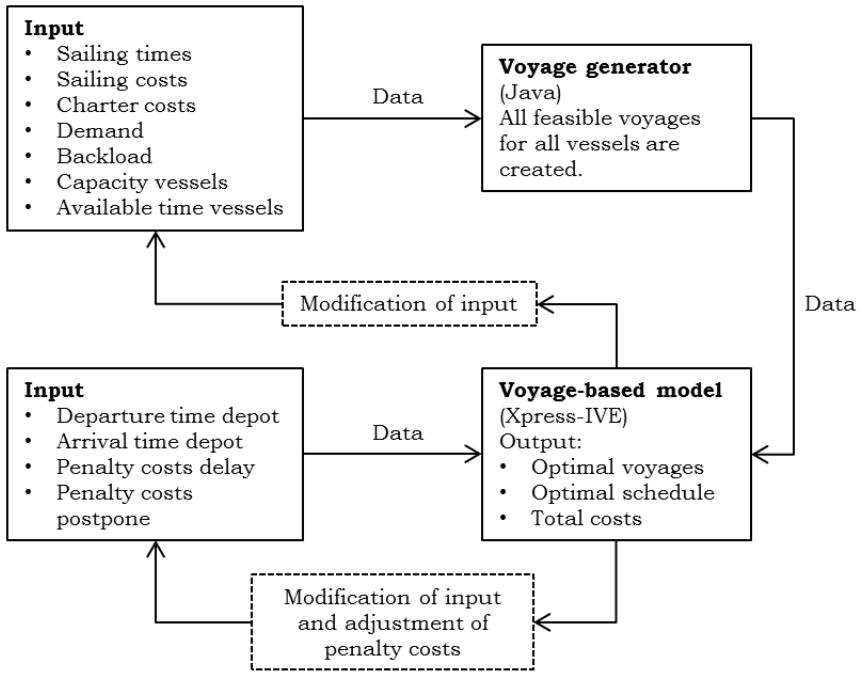

Figure 6.1: Schematic overview for the voyage-based solution method.

6.1 Voyage-based formulation

In this section the voyage-based formulation of the problem is presented, including the notation that differs from the one used in the arc-flow formulation.

The set $\mathcal{R}_v$ contains all possible voyages for vessel v, where each voyage r is feasible with respect to capacity, flow, and time restrictions. If node i is included in voyage r, the parameter A_{ri} equals 1, and 0 otherwise.

The cost associated with sailing and servicing all nodes in a voyage r for OSV v is denoted by C^S_{vr}. In order to reduce the number of constraints in the voyage-based formulation, the time charter costs C^{TC}, in addition to the fixed charter costs, are added to $C^S_{(k+1)r}$ for all voyages r. Thus, the constraint (5.17) is not needed in the voyage-based formulation.

The parameter T^S_{vr} denotes the time needed for sailing and servicing all nodes in a voyage r for OSV v.

The variable x_{vr} equals 1 if OSV v sails voyage r, and 0 otherwise. As in the arc-flow model, if the order at node i is postponed, the variable u_i equals 1, and 0 otherwise. The number of hours OSV v arrives at the depot after planned arrival time T^{MAX}_v on the next voyage is represented by the variable t^D_v.

Objective

$$min \sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{R}_v} C^S_{vr} x_{vr} + \sum_{i \in \mathcal{N}} C^R_i u_i + \sum_{v \in \mathcal{V} \setminus \{k+1\}} C^D_v t^D_v \tag{6.1}$$

The objective function (6.1) consists of three parts. The first part summarizes the costs related to sailing the considered voyages for all OSVs. The second part summarizes the costs associated with orders that are postponed until a later voyage. The third part summarizes the costs related to OSVs that return to the onshore supply depot after planned arrival.

Sailing and service constraints

$$\sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{R}_v} A_{ri} x_{vr} + u_i = 1 \qquad i \in \mathcal{N} \setminus \{0, n+1\} \tag{6.2}$$

$$\sum_{r \in \mathcal{R}_v} x_{vr} \leq 1 \qquad v \in \mathcal{V} \tag{6.3}$$

Constraints (6.2) ensure that all orders are either serviced by an OSV or postponed until a later voyage. Constraints (6.3) ensure that each OSV sails at most one voyage.

Time constraints

$$T_v^{MIN} + \sum_{r \in \mathcal{R}_v} T_{vr}^S x_{vr} - T_v^{MAX} \leq t_v^D \qquad v \in \mathcal{V} \setminus \{k+1\} \tag{6.4}$$

Constraints (6.4) sets the delay variable for each OSV in the long-term fleet.

Binary and non-negativity constraints

$$x_{vr} \in \{0, 1\} \qquad v \in \mathcal{V}, r \in \mathcal{R}_v \tag{6.5}$$

$$u_i \in \{0, 1\} \qquad i \in \mathcal{N} \tag{6.6}$$

$$t_v^D \geq 0 \qquad v \in \mathcal{V} \setminus \{k+1\} \tag{6.7}$$

Constraints (6.5) - (6.7) are the binary and non-negativity constraints.

6.2 Voyage generation using dynamic programming

Similar to the approach in Dell'Amico et al. (2006), pregeneration of all feasible voyages $\mathcal{R}_v$ for each OSV v is performed by using dynamic programming. All voyages are generated for each OSV v through $|\mathcal{N}|$ *stages*, where $|\mathcal{N}|$ is the number

of nodes in the network. The chosen approach applies full enumeration of possible paths with removal of infeasible and dominated voyages.

6.2.1 Label data

In each stage, new states are created. Each *state* is represented by a label containing the following data:

- i - The current node
- R - The predecessor label
- V - The integer number denoting the nodes visited
- C - The sailing and service cost
- T - The sailing and service time
- π^D - The deck capacity needed for deliveries and backload
- π^P - The deck capacity needed for backload

A complete label is written

$$S = \{i,\ R,\ V,\ C,\ T,\ \pi^D,\ \pi^P\}.$$

Following, the current node i for state S is denoted by $i(S)$. Similar notation is used for the rest of the label data, that is $R(S)$, $V(S)$, $C(S)$, $T(S)$, $\pi^D(S)$ and $\pi^P(S)$.

The nodes visited in each state is represented by an integer number V. When a node i is visited, 2^i is added to the number V. That is, node i is visited in S if

$$V \ modulo\ 2^{i+1} \geq 2^i.$$

To find the succession of the nodes in a path represented by V for the state S, all predecessors of S with associated current nodes, are identified.

The initial stage contains one initial state

$$S_0 = \{0,\ \emptyset,\ 2^0,\ 0,\ 0,\ 0,\ 0\},$$

which is the state representing an OSV starting at the depot pickup node. The initial state has no predecessor, which is denoted by $\emptyset$.

6.2.2 Label extension

When extending a label along an arc $(i(S),\ j)$, a new state S' is created at node j. The label data are updated as follows:

$$i(S') = j \tag{6.8}$$

$$R(S') = S \tag{6.9}$$

$$V(S') = V(S) + 2^j \tag{6.10}$$

$$C(S') = C(S) + C^S_{vi(S)j} \tag{6.11}$$

$$T(S') = T(S) + T^S_{vi(S)j} \tag{6.12}$$

$$\pi^D(S') = \begin{cases} max\{\pi^D(S) + D_j, \pi^P(S)\},\ \text{if } j \in \mathcal{N}^D \\ max\{\pi^D(S), \pi^P(S) + P_j\},\ \text{if } j \in \mathcal{N}^P \end{cases} \tag{6.13}$$

$$\pi^P(S') = \begin{cases} \pi^P(S),\ \text{if } j \in \mathcal{N}^D \\ \pi^P(S) + P_j,\ \text{if } j \in \mathcal{N}^P \end{cases} \tag{6.14}$$

Equations (6.8) and (6.9) update the current node and the predecessor for S'. The new current node is marked as visited in equation (6.10). The cost and time data are updated in equations (6.11) and (6.12). The capacity data are updated in equations (6.13) and (6.14) according to whether node j is a delivery or a pickup node.

Figure 6.2 illustrates extension of states in a network with four nodes. An example showing how the capacity data are updated along a path is given in Figure 6.3.

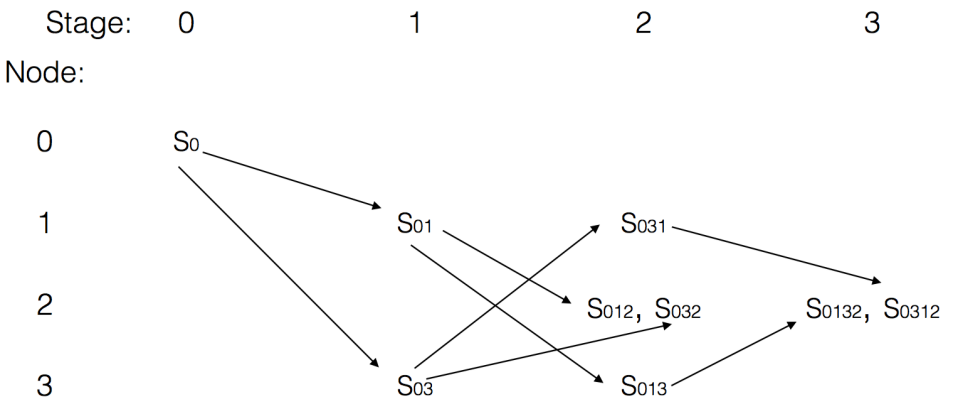

Figure 6.2: Illustration of label extension with 4 nodes. The paths are denoted by the subscript of each state S. All paths starts in node 0 and all final states end in node 2, which are the depot pickup and delivery nodes, respectively.

Let the maximum load on voyage r be carried on arc $(i^*,\ j^*)$. Then the maximum load on r is

$$l_{i^*j^*} = \sum_{j \in \Omega^P} P_j + \sum_{j \in \Theta^D} D_j,$$

where $\Omega^P \subseteq \mathcal{N}^P$ is the set of pickup nodes already visited and $\Theta^D \subseteq \mathcal{N}^D$ is the set of delivery nodes not yet visited.

When the final state S_f is associated with r, $\pi^D(S_f)$ equals the maximum load carried by the OSV during the voyage.

Proposition 6.1.

$$\pi^D(S_f) = \sum_{j \in \Omega^P} P_j + \sum_{j \in \Theta^D} D_j,$$

that is, $\pi^D(S_f)$ equals the maximum load carried during voyage r.

Proof.

For each state with current node i,

$$\pi^D = max\{\pi^D + D_i, \pi^P + P_i\},$$

where $D_i = 0$ if $i \in \mathcal{N}^P$ and $P_i = 0$ if $i \in \mathcal{N}^D$. Whenever $\pi^D = \pi^P$, which implies that the accumulated pickup so far is greater or equal to the accumulated deliveries so far,

$$\pi^D = \sum_{j \in \Omega^P} P_j.$$

Thus, only the pickups before $(i^*,\ j^*)$ is considered, and only the deliveries after $(i^*,\ j^*)$.

Consider the states S_1 and S_2 with current nodes i_1 and i_2, respectively. S_1 is the predecessor of S_2, and both states are predecessors of S_f. Let $\mathcal{N}_f^D$ denote the set of delivery nodes in the path of S_f. Assume that the nodes contained in r are numbered after the sequence in which they are visited. Then, the load on deck after visiting node i_1 and i_2, denoted by l_{i_1} and l_{i_2}, respectively, is

$$l_{i_1} = \sum_{j \in \mathcal{N}_f^D} D_j - \sum_{0 \le j \le i_1} D_j + \sum_{0 \le j \le i_1} P_j, \text{ and} \tag{6.15}$$

$$l_{i_2} = \sum_{j \in \mathcal{N}_f^D} D_j - \sum_{0 \le j \le i_2} D_j + \sum_{0 \le j \le i_2} P_j. \tag{6.16}$$

Assume that either l_{i_1} or l_{i_2} is the maximum load on r. The difference in load is

$$l_{i_1} - l_{i_2} = \sum_{i_1 < j \le i_2} P_j - \sum_{i_1 \le j \le i_2} D_j \tag{6.17}$$

If $l_{i_1} > l_{i_2}$, then

$$\pi^D(S_2) = max\{\pi^D(S_1) + \sum_{i_1 \le j \le i_2} D_j, \pi^P(S_1) + \sum_{i_1 < j \le i_2} P_j\}$$

$$= \pi^D(S_1) + \sum_{i_1 \le j \le i_2} D_j, \text{ and}$$

$$l_{i^* j^*} = \sum_{j \in \mathcal{N}_f^D} D_j - \sum_{0 \le j \le i_1} D_j + \sum_{0 \le j \le i_1} P_j$$

If $l_{i_2} > l_{i_1}$, then

$$\pi^D(S_2) = max\{\pi^D(S_1) + \sum_{i_1 \leq j \leq i_2} D_j, \pi^P(S_1) + \sum_{i_1 < j \leq i_2} P_j\}$$

$$= \pi^P(S_1) + \sum_{i_1 < j \leq i_2} P_j, \text{ and}$$

$$l_{i^*j^*} = \sum_{j \in \mathcal{N}_f^D} D_j - \sum_{0 \leq j \leq i_2} D_j + \sum_{0 \leq j \leq i_2} P_j$$

Not depending on which of l_{i_1} and l_{i_2} is greatest, the maximum load can be expressed as

$$l_{i^*j^*} = \sum_{j \in \Omega^P} P_j + \sum_{j \in \Theta^D} D_j = \pi^D(S_f).$$

Thus, Proposition 6.1 is correct. □

Additional remark; if node i^* were a delivery node, then the load on board before visiting i^* would be greater than $l_{i^*j^*}$. Thus, i^* is a pickup node. Similar, if j^* were a pickup node, then the load on board after visiting j^* would be greater than $l_{i^*j^*}$. Thus, j^* is a delivery node.

An extension of state S to state S' along an arc $(i(S),\ j)$ is feasible if the following inequalities hold:

$$V(S)\ mod\ 2^{j+1} < 2^j \tag{6.18}$$

$$i(S) \neq n+1 \tag{6.19}$$

$$V(S)\ mod\ 2^{j^{Sibling}+1} < 2^{j^{Sibling}},\ \text{if } j \in \mathcal{N}^D \text{ and } j \neq n+1 \tag{6.20}$$

$$T(S) + T^S_{vi(S)j} \leq T_v^{MAX} + \tau - T_v^{MIN},\ \text{if } j = n+1 \tag{6.21}$$

$$T(S) + T^S_{vi(S)j} + T^S_{vj(n+1)} \leq T_v^{MAX} + \tau - T_v^{MIN},\ \text{if } j \neq n+1 \tag{6.22}$$

$$max\{\pi^D(S) + D_j, \pi^P(S)\} \leq Q_v,\ \text{if } j \in \mathcal{N}^D \tag{6.23}$$

$$max\{\pi^D(S), \pi^P(S) + P_j\} \leq Q_v,\ \text{if } j \in \mathcal{N}^P \tag{6.24}$$

$$\pi^P(S) \leq Q_v,\ \text{if } j \in \mathcal{N}^D \tag{6.25}$$

$$\pi^P(S) + P_j \leq Q_v,\ \text{if } j \in \mathcal{N}^P \tag{6.26}$$

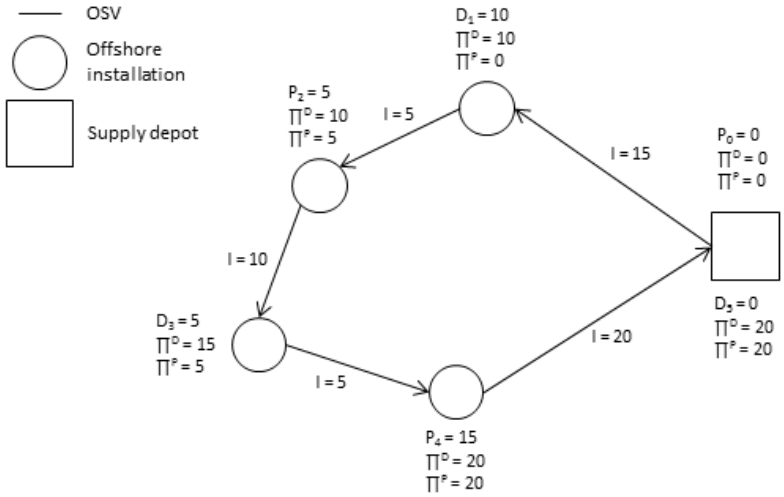

Figure 6.3: Illustration showing how capacity data are updated during label extension. Requested delivery and backload size is given for each delivery and pickup node, respectively. The current amount of cargo carried on deck of the OSV is given for each arc.

If the inequality (6.18) holds, node j is not already visited in the path in state S. No states having the depot delivery node $n+1$ as the current node should be extended. Thus, if the inequalty (6.19) does not hold, S' is infeasible. Inequality (6.20) assures that the pickup node should never be visited before the delivery node for the same installation. A *sibling* of a delivery node is defined as the corresponding pickup node of the installation, if one exists. Likewise, the sibling of a pickup node is the delivery node of the installation. The sibling node of node i is denoted by $i^{Sibling}$. As stated in inequality (6.21), the time needed to service the nodes in a path sailed by OSV v should not exceed the time available to v. As in Chapter 5, τ is the maximum allowed delay on a voyage. If node j is not the depot delivery node, there should also be enough time available to sail back to the onshore supply depot. This is assured by inequality (6.22). The inequalities (6.23) - (6.26) hold if the load on deck at any point of the path does not exceed the available deck capacity of the OSV.

6.2.3 Label domination

At each stage, all states dominated by another state are identified and removed. Label S_1 dominates S_2 if all feasible extensions of S_2 are feasible for S_1, and if extending S_2 to a set of nodes never give a better solution than extending S_1 to the same set of nodes. The label dominance criteria are:

Proposition 6.2. *The label S_1 dominates S_2 if*

$$V(S_1) = V(S_2), \tag{6.27}$$

$$i(S_1) = i(S_2), \tag{6.28}$$

$$C(S_1) \leq C(S_2), \tag{6.29}$$

$$T(S_1) \leq T(S_2), \tag{6.30}$$

$$\pi^P(S_1) \leq \pi^P(S_2), \tag{6.31}$$

$$\textit{and } \pi^D(S_1) \leq \pi^D(S_2). \tag{6.32}$$

Proof.

Criterion (6.27) assures that S_1 and S_2 visit the same set of nodes. Also, (6.28) assures that S_1 and S_2 end at node

$$i = i(S_1) = i(S_2).$$

Consider the partial path p' starting at i and ending at the depot delivery node $n+1$. Let p_1 and p_2 be the complete paths obtained when extending S_1 and S_2 with p', respectively. The cost of the (partial) path p is denoted $C(p)$. The costs associated with sailing and servicing nodes are separable and independent of the nodes already visited. Therefore,

$$C(S_1) \leq C(S_2) \implies C(S_1) + C(p') \leq C(S_2) + C(p') \implies C(p_1) \leq C(p_2).$$

Thus, criterion (6.29) is correct.

Similarly, the time needed for the (partial) path p is denoted $T(p)$. The time needed for sailing and servicing nodes is separable and independent of the nodes

already visited. Therefore,

$$T(S_1) \leq T(S_2) \implies T(S_1) + T(p') \leq T(S_2) + T(p') \implies T(p_1) \leq T(p_2),$$

and

$$T(p_2) \leq T_v^{MAX} + \tau - T_v^{MIN} \implies T(p_1) \leq T_v^{MAX} + \tau - T_v^{MIN}$$

With respect to time, extending S_2 never gives a better solution than extending S_1 with the same partial path, and all extensions of S_2 are feasible for S_1. Thus, criterion (6.30) is correct.

Let the accumulated backload of the nodes in the (partial) path p be denoted $\pi^P(p)$. The capacity needed for carrying $\pi^P(p')$ is independent of the nodes visited before p'. Therefore,

$$\pi^P(S_1) \leq \pi^P(S_2) \implies \pi^P(S_1) + \pi^P(p') \leq \pi^P(S_2) + \pi^P(p')$$
$$\implies \pi^P(p_1) \leq \pi^P(p_2),$$

and

$$\pi^P(p_2) \leq Q_v \implies \pi^P(p_1) \leq Q_v.$$

With respect to the capacity needed for pickups, all extensions of S_2 are feasible for S_1. Thus, criterion (6.31) is correct.

The maximum needed capacity at the (partial) path p is denoted $\pi^D(p)$.

$$\pi^D(p_1) = max\{\pi^D(S_1) + \sum_{j \in \Theta^D} D_j, \sum_{j \in \Omega^P} P_j + \pi^D(p')\}$$
$$\pi^D(p_2) = max\{\pi^D(S_2) + \sum_{j \in \Theta^D} D_j, \sum_{j \in \Omega^P} P_j + \pi^D(p')\},$$

where Θ^D is the set of delivery nodes visited in p', and Ω^P is the set of pickup nodes visited in p_1 and p_2.

Since,

$$\pi^D(S_1) \leq \pi^D(S_2) \implies \pi^D(p_1) \leq \pi^D(p_2),$$

$$\pi^D(p_2) \leq Q_v \implies \pi^D(p_1) \leq Q_v.$$

With respect to the capacity constraints, all extensions of S_2 are feasible for S_1. Thus, criterion (6.32) is correct.

All the criteria are proven to be correct, thus Proposition 6.2 is correct. □

If $i(S_1)$ and $i(S_2)$ are the depot delivery node $n+1$, S_1 dominates S_2 given only criteria (6.27) - (6.30). Criteria (6.31) and (6.32) are only needed to make sure that all feasible extensions of S_2 are feasible for S_1. However, no final states are extended, and therefore, criteria (6.31) and (6.32) are not needed when dominating final state labels.

The sequence in which the dominance criteria are performed is of significance for the computational time of the algorithm. It is preferable to identify dominated states as early as possible. Most of the states will not have visited the same set of nodes, and therefore, the dominance criterion (6.27) is performed first. Similarly, many of the states will not have the same current node. Consequently, (6.28) is the second dominance criterion. The sequence of the remaining dominance criteria (6.29) - (6.32) is not considered as significant to the computational time.

6.2.4 Pseudocode for voyage generation

Algorithm 6.1 shows simplified pseudocode for the generation of voyages. All states are generated simultaneously for all OSVs. The feasibility of each state is determined by Algorithm 6.2, which returns a list with an element for each OSV v, stating whether the state is feasible for v or not. If the state is not feasible for any of the OSVs, the state is considered infeasible. If the state is feasible for at least one of the OSVs, the state is considered feasible and is extended or added to the set of final states for at least one of the OSVs. Each OSV has one associated set of final states.

For each new state, Algorithm 6.3 checks whether the state dominates any of the other states in the current stage, and whether the new state is dominated by any

of the other states. If the new state is dominated by another state, the new state will not dominate any existing states that are not already dominated. Therefore, Algorithm 6.3 terminates as soon as a new state is identified as dominated, and Algorithm 6.1 proceeds.

Algorithm 6.1 Pseudocode for dynamic voyage generation

```
1:  procedure VOYAGEGENERATOR
2:      Create initial state
3:      Add initial state to initial stage
4:      k = 1
5:      while k < |N| do
6:          for all states in stage M_k do
7:              for all nodes i in N do
8:                  if i is visited or
                    the depot delivery node is visited or
                    (i is a delivery node and the pickup node for the
                    same installation is visited) then
9:                      Next iteration
10:                 else if i is depot delivery node then
11:                     Create new final state
12:                 else
13:                     Create new state
14:                 end if
15:                 if the new state is feasible for one or more OSVs
                    (Algorithm 6.2) then
16:                     Add the new state to M_{k+1}
17:                     Find all states in M_{k+1} dominated by the new state
18:                     or any state dominating the new state (Algorithm 6.3)
```

```
            end if
        end for
    end for
    k = k + 1
    Remove all dominated states from M_k
    Add all new final states to the set of final states for each OSV
end while
Return all final states
end procedure
```

Algorithm 6.2 Pseudocode for the feasible state procedure

```
procedure FEASIBLESTATE
    for all OSVs v do
        if current node is depot delivery node and
        the time used exceeds the time available then
            State not feasible for OSV v
        else if the time used plus the time needed to sail back
        to the depot exceeds the time available then
            State not feasible for OSV v
        else if needed delivery capacity > Q_v then
            State not feasible for OSV v
        else if needed pickup capacity > Q_v then
            State not feasible for OSV v
        else
            State feasible for OSV v
        end if
    end for
    Return feasibility for all OSVs
end procedure
```

Algorithm 6.3 Pseudocode for the state domination procedure

```
1: procedure DOMINATEDSTATES(State S, Stage M_k)
2:     for all states R in M_k except S do
3:         if S and R visit different nodes or
           S and R have different current nodes then
4:             Next iteration
5:         end if
6:         if S dominates R and R is not already dominated then
7:             Dominate R
8:             Next iteration
9:         else if R dominates S then
10:            Dominate S
11:            Return S
12:        else
13:            Next iteration
14:        end if
15:    end for
16:    Return dominatedstates
17: end procedure
```

Chapter 7

A variable neighborhood search heuristic

As the focus of this report is on operational planning and disruption management, it is essential that the model used to solve the problem can provide good solutions within a reasonable amount of time. In several of the papers mentioned in Chapter 3, the authors concluded that exact solution methods can provide optimal solutions within reasonable time for small instances of pickup and delivery problems. However, the solution time increases exponentially with the problem size. Thus, larger instances are computationally difficult to solve with exact methods, such as those presented in Chapter 5 and Chapther 6. Furthermore, computational memory imposes a restriction on how large instances the voyage-based method is able to generate all voyages for. For instances that are too large for the exact methods to handle, an alternative method is needed to provide good solutions in relatively short time. A variable neighborhood search (VNS) heuristic for solving the problem is therefore proposed. The outline of this chapter is as follows; in Section 7.1, the VNS is described at an aggregate level. In Section 7.2, the construction of initial solutions is explained. In Section 7.3, the criteria for evaluating a solution

and methods for rearranging it are described. In Section 7.4, the local search and the neighborhood operators are presented. Finally, in Section 7.5, a mechanism for perturbing the solution is explained.

7.1 Heuristic overview

In this section, an overview of the proposed heuristic is given, which is a variable neighborhood search (VNS) with perturbations. A *perturbation* consists of destroying parts of the solution and rebuilding it in order to escape local optima.

Algorithm 7.1 shows, at an aggregate level, how the heuristic works. In this algorithm, the outer loop controlled by the n_{max} parameter is referred to as the *rearrangement phase*, while the inner loops controlled by the p_{max} and s_{max} parameters are referred to as the *improvement phase*.

First, a construction heuristic is run to construct an initial solution which is used as input for the VNS heuristic. How this is done is described in detail in Section 7.2. Then, the rearrangement and improvement of the solution starts. The number of times the improvement phase is run, is controlled in the rearrangement phase, which is described in Section 7.3. In the rearrangement phase, the solution is either rearranged, or accepted based on which criteria that are met in Algorithm 7.3.

Shaking and local search is run to improve the current best found solution in the improvement phase. *Shaking* is a way to slightly disturb the current solution and obtain a new starting point for the local search. This is done by performing a neighborhood move and using the result as input for the local search. A solution is an improvement if it has a lower total cost than the current best found solution, and the constraints regarding deck capacity and total voyage duration are not violated. In Section 7.4, the shaking and local search are further described. After the shaking and local search is done, a check is performed to see if an improved solution can be found by *reassigning* the voyages between the vessels, since this

might reduce delays. Then, the solution is *changed* so that delivery nodes are visited before pickup nodes belonging to the same installations if they are both visited consecutively by the same vessel. This is done, since Proposition 5.1 states that visiting the pickup node before the delivery node for the same installation never gives a better solution than vice versa. If the shaking and local search fails to find an improved solution after s_{max} iterations, a perturbation is performed. A detailed description of the perturbation is given in Section 7.5.

Algorithm 7.1 Pseudocode for the VNS heuristic

```
1:  procedure VNS
2:      Construct an initial solution (Algorithm 7.2)
3:      for n = 0 to n_max do
4:          Do a solution check and rearrange (Algorithm 7.3)
5:          for p = 0 to p_max do
6:              for s = 0 to s_max do
7:                  run shaking (Algorithm 7.4) and local search (Algorithm 7.5)
8:                  Reassign voyages and change sibling order
9:                  if the new solution is better than the best found solution then
10:                     Update the best found solution
11:                     p ← 0
12:                     s ← 0
13:                 end if
14:             end for
15:             perturbate the best found solution (Algorithm 7.6)
16:         end for
```

```
        if the new solution is better than the best found solution then
            Update the best found solution
        end if
    end for
    Return the best found solution
end procedure
```

7.2 Initial solution

In Algorithm 7.2, pseudocode of the procedure for finding the initial solution for the VNS heuristic is presented. As in the multi-start local search heuristic presented in Brønmo et al. (2007), the procedure for constructing an initial solution is based on finding several solutions and picking the best one as the initial solution.

The following is done $i_{max}/2$ number of times to create one new solution each time. The algorithm starts by making sure all vessels have the depot pickup and delivery node as their start and end node, respectively. Then, a random vessel and a random delivery node is picked. The vessel might be an OSV in the current fleet or a spot vessel. The delivery node is added to the vessel's route if it gives a feasible solution. In a feasible solution, the vessel capacity and time restrictions are not violated. Then, the delivery node's sibling, if one exists, is added after the mentioned node if it also gives a feasible solution. As mentioned in Section 6.2.2, *sibling nodes* are defined as a pickup node and a delivery node belonging to the same installation. This is continued until the all delivery nodes are added, i.e. the *delivery nodes list* is empty, or no nodes are added because of infeasibility for m iterations. Then, the remainders in the *pickup node list* are added in the same way. If there exist nodes which are not added to the solution, these are added to the solution's *postponed nodes list*. Afterwards, the same procedure is repeated $i_{max}/2$ number of times, but this time nodes are only added to OSVs in the long-term fleet, not the spot vessel. This is done to help the heuristic find solutions where the

spot vessel is not used, since this might be the solution with lowest cost, especially for instances with no disruptions.

Thus, in Algorithm 7.2, i_{max} solutions are created and every solution is saved in a list $\mathcal{L}$. In the end of Algorithm 7.2, $\mathcal{L}$ is iterated, and the solution with lowest cost is returned. This approach is quite similar to how the initial solution in the tabu search heuristic in Korsvik et al. (2009) is constructed. Here, cargoes are assigned to randomly selected vessels, but unlike in Algorithm 7.2, the vessel capacity and time constraints may be violated.

Algorithm 7.2 Pseudocode for constructing an initial solution

1: **procedure** INITIAL SOLUTION
2: $\mathcal{L}$ is a list of initial solutions
3: $counter = 0$
4: **for** $i = 0$ to $i_{max}/2$ **do**
5: Create a new solution and add the depot in the start and end of to every vessel's voyage.
6: **while** *delivery nodes list* is not empty **and** $counter < m$ **do**
7: Add a random delivery node to a random vessel's voyage and remove it from the *delivery nodes list* if it gives a feasible solution. Then, also add the node's sibling to the vessel's voyage and remove it from the *pickup nodes list* if it gives a feasible solution. If no nodes are added, increase *counter* by 1.
8: **end while**
9: $counter = 0$
10: **while** *pickup nodes list* is not empty **and** $counter < m$ **do**
11: Add a random pickup node to a random vessel's voyage and remove it from the *pickup nodes list* if it results in a feasible solution. If no node is added increase *counter* by 1.
12: **end while**

13: **if** some nodes are not added to any vessel's voyage **then**
14: Add these nodes to the solution's *postponed nodes list*.
15: **end if**
16: Add solution to $\mathcal{L}$
17: **end for**
18: Repeat line 3-17, but this time only add nodes to OSVs on long term contract.
19: Find the solution with lowest cost in $\mathcal{L}$, including the cost of postponing nodes, and return this solution.
20: **end procedure**

7.3 Check and rearrange

Postponing orders is a possible option for reducing the delay on a vessel's return time to the onshore depot. Delays might propagate in the weekly schedule and cause significant costs for Statoil. Furthermore, in the cases of increased loads, it might not be possible to service all installations within the planning period due to capacity restrictions on the vessels, and some orders might be postponed. The rearranging of a solution is based on which criteria the solution fulfills. If there are delays, the following two possibilities exist:

- **Placing two sibling nodes that were separated, together.** Siblings are *separated* if they are placed on different vessels or if one of them is postponed. If the siblings are visited on the same voyage, but not subsequently, they are not considered as separated. If the heuristic has not tried to move a separated node before, according to a tabu list, it is moved to its sibling. This is done until all the separated siblings have been tried to be reunited. This rearrangement is denoted *sibling reunion* in Algorithm 7.3.

- **A sibling node pair can be moved to another vessel.** This is done by checking all voyages and finding the node that is furthest away from all other

nodes in its voyage. That node is moved along with its sibling, regardless of where the sibling is located. They are moved to a voyage that is not the one where the node furthest away was located. The voyage chosen is where the distance from the sibling pair to all other nodes in that voyage is the shortest. A tabu list is used to keep track of which nodes have been tried to be moved, so no nodes are being moved more than one time. This rearrangement is denoted *Move sibling pairs* in Algorithm 7.3.

If the solution contains postponed nodes, the following two possibilities exists

- **Exchanging a node that is postponed with a node that is currently in a voyage.** This is done by finding the non-postponed node with the highest load, and exchanging its place with the postponed node with the lowest load. Since a node with high load is replaced by a node with low load the total load of the voyage can be reduced. This can open for the possibility of reinserting more postponed nodes with low loads to the voyages later. This might lead to fewer postponed nodes, which in this model will most likely result in a lower objective value. This is tried until 0.4 x n_{max} unsuccessful exchanges have taken place, and a tabu list is used to not exchange the same nodes more than one time. This rearrangement is denoted *Drop and reinsert* in Algorithm 7.3.

- **Reinserting a postponed node.** All positions in all voyages are tried, and the best position that is also feasible, is chosen. This rearrangement is denoted *reinsert* in Algorithm 7.3. If no feasible position is found, no more reinserting is done, and the heuristic finishes. Also here, a tabu list is used to ensure that all nodes are being tried reinserted exactly one time.

If the solution is neither delayed nor has any postponed nodes, the improvement phase of Algorithm 7.1 is run three times, this is ensured in line 21 in Algorithm 7.3. Here, the criteria for which rearrangement that is chosen is also shown.

Algorithm 7.3 Pseudocode for checking and rearranging a solution

```
1: procedure SOLUTION CHECK AND REARRANGE
2:     if n > 0 and the best found solution has delay then
3:         Do sibling reunion
4:         if all nodes has been tried in sibling reunion then
5:             Move sibling pairs a maximum of n_max/2 times
6:         end if
7:     else if n > 0 and any nodes are postponed in the best found solution then
8:         Drop and reinsert
9:         if an improved solution is found then
10:            n ← 1
11:        else if an improvement is not found after 0.4 x n_max attempts then
12:            Try to reinsert a postponed node if it is not in the tabu list, and add
               it to the tabu list if one is found.
13:            if n = n_max and all nodes have not been tried reinserted then
14:                n ← n − 1
15:            end if
16:            if no node can be feasibly reinserted then
17:                return the best found solution
18:            end if
19:        end if
20:    else
21:        n_max ← 2
22:    end if
23: end procedure
```

7.4 Shaking and local search

The shaking and local search use a VNS structure by changing the neighborhood operator used to explore the solution space. The use of shaking to generate new starting points for the local search was chosen due to the similar implementation described in Polat et al. (2015), which gave high quality solutions for a VRP with simultaneous pickup and delivery. The shaking decides a search direction, and the solutions obtained by the shaking are further evaluated using a local search in order to explore new promising neighborhoods of the current solution.

Since a local optima for one neighborhood structure might not be optimal for another neighborhood structure, a larger part of the solution space is searched before a local optima is reached, than what would have been explored with a single neighborhood structure. When a local optima is reached with respect to all neighborhoods defined, the solution is perturbed.

The implementation described in this report employs nine neighborhood structures - four *intra-route* operators that move nodes within a given route, and five *inter-route* operators that move nodes between two different routes. In the following, *move* and *operator* are used interchangeably when referring to the neighborhood structures described below. The pseudocode for the shaking is given in Algorithm 7.4, and the neighborhood structures employed are shown in Figure 7.1 and subsequently explained below. Many of the moves described are the same as the ones described in Caseau and Laburthe (1999), Brønmo et al. (2007), Korsvik et al. (2009), and Polat et al. (2015). The implementation of the moves described in this report uses a large degree of randomness in the selection of which vessels, nodes, and positions that are used in each move.

The intra-route operators used are *2-opt*, *3-opt*, *swap* and *insert*:

- The *2-opt* move takes two random nodes from a voyage, removes one edge from each of these that are not neighboring edges, reconnects the graph, and switches the direction of visit of the intermediate nodes. In Figure 7.1a), the

edges (1,2) and (4,5) are deleted and replaced with (1,4) and (2,5). Note that the direction of visits for nodes 2 through 4 are reversed.

- *3-opt* is similar to the 2-opt, but instead three random nodes are selected, one edge is removed from each node (again, the edges are not neighboring), and the graph is reconnected. The intermediate nodes between two of the chosen nodes has their direction of visit reversed. In Figure 7.1b) the edges (0,1), (2,3) and (5,6) are deleted and replaced with (0,5), (3,1) and (2,6). Similar to the 2-opt case, the direction of visit for nodes 3 through 5 are reversed.

- *Swap* takes two random nodes in a voyage, and swaps their location in the sequence of visits. In Figure 7.1c), nodes 3 and 4 are swapped.

- The *insert* operator takes a random node, removes it from a voyage, and inserts it in another random place in the same voyage. In Figure 7.1d), node 4 is inserted between nodes 1 and 2.

The inter-route operators are *exchange*, *cross*, *shift*, *replace* and the operator the authors have called *sibling mover*.

- *Exchange* takes a random number m of sequential nodes from one voyage and exchanges them with n nodes from another voyage, which is randomly chosen to be either m or *m-1*. In Figure 7.1e), m is set to 2 and n to 1, and nodes 1 and 2 from the first voyage is exchanged with node 7 node from the second voyage.

- The *cross* operator works by deleting the edges between two consecutive nodes in two different voyages, and reconnecting the voyages with each other. In Figure 7.1f), edges (2,3) and (5,6) are deleted and replaced with (2,6) and (5,3)

- *Shift* removes a random node from a voyage, and inserts it into a random location in another voyage. In Figure 7.1g), node 1 is taken from the first

voyage and inserted between nodes 5 and 6 in the second voyage.

- *Replace* takes one node randomly from two different voyages and exchanges their locations. In Figure 7.1h), nodes 1 and 6 are exchanged.

- *Sibling mover* does the same as the sibling mover explained in Section 7.3 only with a few modifications. One of the differences is that a random vessel is chosen instead of investigating all vessels. The node furthest away from the other nodes within this vessel's voyage is chosen as the node to be moved. As in Section 7.3 the node's sibling is also moved (if it exists) regardless of which vessel it is currently serviced by. The reinsering of the sibling pair is conducted the same way as in Section 7.3. *Sibling mover* is not included in Figure 7.1 since it can be seen as a sophisticated *Shift* (possibly shifting two nodes instead of one). Even though almost the same procedure is done in Algorithm 7.3, the authors think it is such an important move that it also should be done in the shaking and local search. One important thing to notice is that *Sibling mover* is only done if some of the vessels in the initial solution are delayed. This is because *Sibling mover* uses the sailing times to decide which node to move, and this is most applicable for instances where the vessels are delayed.

The neighborhood moves described above (except *sibling mover*) are done in two different manners depending on whether the move is performed by an intra-route or an inter-route operator. For intra-route moves it is done by selecting a random vessel and performing the move corresponding to the neighborhood structure. For inter-route moves, two different vessels are randomly selected and the move is performed on the voyages of the two vessels.

The aim of the shaking is as mentioned earlier to provide a search direction and thus obtain a new starting point for a local search, and help escape some of the local optima. This operation is similar to that described in Polat et al. (2015) since the shaking is integrated with the local search.

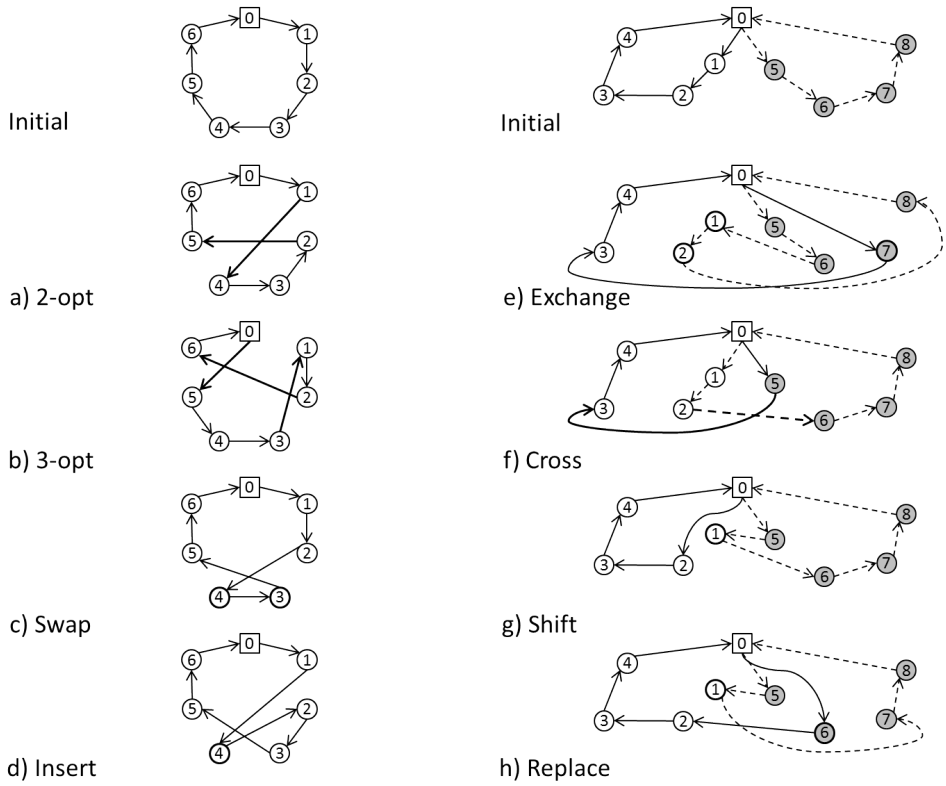

Figure 7.1: Neighborhood structures used in shaking and local search. Left: intra-route operators. Right: inter-route operators.

Algorithm 7.4 Pseudocode for the shaking

```
1: procedure SHAKING
2:     for k = 0 to k_max do
3:         Find a solution in the k^th neighborhood of the current solution using
           an inter-route or intra-route operator
4:         if the new solution is better than the best found solution then
5:             Update the best found solution
6:             k ← 0
7:         else
8:             run local search (Algorithm 7.5)
9:             if the new solution from Algorithm 7.5 is better than the best found
               in this run then
10:                Update the best found solution
11:                k ← 0
12:            end if
13:        end if
14:    end for
15:    Return the best found solution
16: end procedure
```

First in the shaking in Algorithm 7.4, a solution in one of the k neighborhoods of the *current solution* is found by use of one of the neighborhood operator described above. The neighborhood operators are numbered from zero to eight, e.g. if $k = 5$, the neighborhood operator numbered five is chosen. If the neighborhood operator that was tried in the shaking step results in an improved solution, the move is performed, the best found solution is updated, and the shaking starts with its first operator again. If the move is non-improving, the move is still performed, and a local search, described in Algorithm 7.5, is run on the solution obtained by performing the move. This is done to further explore the neighborhood for

promising solutions. The solution returned from the local search is checked against the best found solution in the shaking, and if the moves from the local search are improving, the moves are performed, the best found solution is updated, and the shaking restarts from its first operator. Otherwise, the shaking continues with its next operator.

The local search uses the same neighborhood operators in the same order as the shaking, but the local search never starts from the first operator if an improved solution is found. The local search employs the moves sequentially, updates the solution after an improving move is performed, and returns the best solution it finds.

Algorithm 7.5 Pseudocode for the local search

```
1: procedure LocalSearch
2:     for m = 0 to m_max do
3:         Find a solution in the m^th neighborhood of the current solution using
           an inter-route or intra-route operator
4:         if the new solution is better than the best found solution
           then
5:             Update the best found solution
6:         end if
7:     end for
8:     Return the best found solution
9: end procedure
```

7.5 Perturbation - destroy and repair

The shaking and local search might get stuck in a local optima. To escape a local optima, a perturbation of the current solution is needed. A methodology which is based on the ejection chain described in Subramanian, Drummond, Bentes, Ochi, and Farias (2010) is therefore proposed. Pseudocode for the perturbation is given

in Algorithm 7.6.

The ejection chain works by selecting a random node from the first vessel in the solution, if it has any nodes other than the depot nodes. If the sibling of the chosen node is on the same vessel, it is also selected. The node(s) are then removed from the chosen vessel, and inserted into the next vessel in the solution. Sibling nodes are moved together since they in most real life cases would be visited at the same time. The procedure continues with the remaining vessels, and for the last vessel in the solution, any removed nodes are inserted into the first vessel. The ejection chain is illustrated in Figure 7.2.

This procedure can result in infeasible solutions since the voyage duration and capacity constraints are not checked in the perturbation. However, it might give the opportunity of exploring new areas of the search space that have not previously been visited. It is important to ensure that the degree of the solution that is destroyed is not too low or too high. A low ratio might not cause large enough parts of the solution to be changed, resulting in unexplored areas of the solution space not being searched. A high ratio might cause the solution to change so much that the model end up re-optimizing instead of improving. Algorithm 7.6 ensures that parts of the solution are changed, but the main structure is not altered.

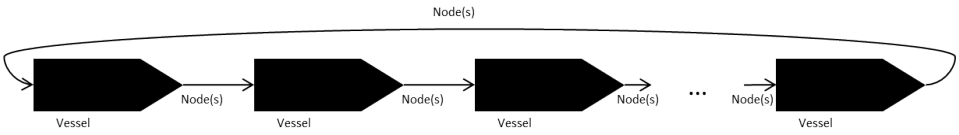

Figure 7.2: Illustration of how the ejection chain works.

Algorithm 7.6 Pseudocode for the perturbation

```
1: procedure EJECTION CHAIN
2:     for i = 0, i ≤number of vessels do
3:         if vessel i contains nodes other than depot then
4:             Remove a random node from the vessel
5:             if the sibling is on the same vessel then
6:                 Remove the sibling from the vessel
7:             end if
8:         else
9:             Continue to next vessel
10:        end if
11:        if i is the last vessel in the list then
12:            Insert the removed node(s) into the first vessel
13:        else
14:            Insert the removed node(s) into vessel i + 1
15:        end if
16:    end for
17: end procedure
```

Chapter 8

Computational study

In this chapter, the computational study performed with the arc-flow model, the voyage-based model, and the heuristic is presented. First, the test instances and parameter values relevant to these solution methods are described in Section 8.1. In Section 8.2, a summary of the penalty cost and parameter testing is given. A comparison of the solution methods is given in Section 8.3. In Section 8.4, detailed examples of solutions found by the heuristic are examined.

The tests are conducted on a computer running Windows 7 with an Intel i7-3770 3.40 GHz CPU and 16 GB of RAM. All Java code is run in Eclipse Luna (Eclipse.org, 2015) with Java Development Kit 1.7.0. The exact models are implemented in Xpress-IVE 1.24.04 with Xpress-Mosel 3.6.0 and solved with Xpress-Optimizer 21.01.04 (Fico.com, 2015).

8.1 Test instances

The test instances are based on data supplied by Statoil. Each test instance consists of an onshore supply depot and a set of offshore installations. The pickup and delivery requests placed by the offshore installations are represented as pickup and delivery nodes in each of the test instances. The depot is also represented by two

nodes, one pickup node where the vessels pick up the offshore installations' demand when the voyage starts, and one delivery node where the vessels return with the backload from the installations in the end of the voyage.

As of today, each of the onshore supply depots services up to 13 offshore installations, i.e. 28 nodes including the depot. The model is tested with various instances with a small number of installations and close to the current maximum number of installations. After a planned reorganization of the offshore supply service in Statoil, one onshore supply depot (Mongstad) will serve up to 26 offshore installations (i.e. 54 nodes). Therefore, larger instances are presented and tested to see how the heuristic handles more nodes than what is realistic today.

In the problem instances tested, the planning horizon has a duration of two to three days. All offshore installations are not always scheduled to be serviced within this planning horizon. Therefore, some of the problem instances of realistic size do not include all installations associated with a given onshore supply depot.

The parameters used in the test instances are:

- **The OSV fleet size**, which is varying between two and six OSVs, including the spot vessel, depending on the onshore supply depot and how many installations that are to be serviced.

- **The daily cost of chartering an OSV from the spot market**, is calculated by finding the average spot market price, which is NOK 275 000/day. The spot market prices are estimates provided by Statoil, and should be adjusted to current prices when the model is used in real life.

- **The fixed charter cost**, which is the cost associated with preparing a newly chartered spot vessel or a spot vessel that is to be delivered back to the company owning the vessel. Some bulk cargo requires preparation of tanks before it is brought to installations. When a spot vessel is to be returned, the tanks must be cleaned to remove the remains of bulk materials. Based on estimates provided by Statoil, the cost is set at NOK 45 320.

- **The cost for postponing the ordered service to a node**. Each order has a priority that reflects how expensive it is to postpone. Higher priority results in higher postpone costs. For the instances tested, 20% of the installations are assumed to have high priority, while the rest of the installations have normal priority. This penalty cost is further discussed in Section 8.2.

- **The hourly cost of an OSV arriving at the onshore supply depot after its schedule**. This penalty cost is also further discussed in Section 8.2.

- **The amount of deck capacity per order**, which is drawn from a truncated normal distribution with a mean of 100 m^2 per order for outgoing supplies and $80m^2$ for backload. The standard deviation is 50 m^2 and 40 m^2, respectively. The minimum and maximum values of the truncation is 1 m^2 and 250 m^2, respectively.

- **The deck capacity for each OSV**, which is set at 1 000 m^2 for all vessels based on information provided by Statoil.

- **The time needed for sailing between and servicing nodes**. The time needed is calculated with a sailing speed of ten knots, known as the environmental speed of the vessels. The planned service time for each node varies between one and three hours.

- **The costs of sailing between and servicing nodes**. The costs are calculated by assuming a fuel consumption of 12 m^3 per day at environmental speed with the costs of marine gas oil at NOK 10 071 per m^3. The fuel consumption is assumed to be 6 m^3 per day when the vessels service installations or are docked at the onshore supply depot, which is also included in the calculations. All numbers used here are provided by Statoil.

- **The planned departure and arrival times at the depot**. The available time for each vessel is 72 hours or less in the cases without disruptions. The

solutions to the instances without disruptions are used to define the time limits for the instances with disruptions in order to see the delays caused by the disruptions. The available time for the spot vessel is 72 hours both in the cases with and without disruptions.

In the following sections, several parameters are tuned, and the different models are compared. The sets of test instances used for testing are described below:

- In the first part of Section 8.2, a summary of the tests conducted to decide the postpone and delay cost parameters used in the input data is given. The parameters are tested for the three supply depots Mongstad, Florø, and Ågotnes, which service 13, 13, and 12 installations, respectively. The instances have close to the maximum number of installations serviced from the depots today, and are shown in Table 8.1.

- A summary of the heuristic parameter tuning is given in the remainder of Section 8.2. To find different parameters for instances of different sizes, one small, one medium, and one large instance are used, as shown in Table 8.2.

- In Section 8.3, the performance of the proposed solution methods are compared. Since there are limitations on the exact models regarding the size of instances they are able to solve within reasonable time, several small instances are tested in order to compare the performance of the exact methods and the heuristic. In addition, larger instances are tested to see how the heuristic performs on today's maximum number of nodes and future possible planning cases. The instances used in Section 8.3 are shown in Table 8.3.

In Tables 8.1-8.3, the size of the instances are specified. The small (S), medium (M), and large (L) classification is used to set the value of different parameters in Section 8.2. The classification is set based on the number of nodes in the instance.

In addition to the non-disrupted test instances, the models are tested with the following kinds of disruptions:

- **Reduced sailing speed** due to adverse weather conditions. This is done by reducing the speed of each vessel from ten to five knots, and thus, increasing the sailing time. The increase in sailing time will also affect the sailing costs. The ID for these instances will have a superscript "S" for speed, such as M_{22}^{S}.
- **High order volumes**, which often is the case after periods where adverse weather conditions have made offshore supply impossible or too costly. This is done by setting the demand and backload amount to the triple of the size used in the non-disrupted case to ensure that deck capacity becomes a binding constraint. The ID for these instances will have a superscript "L" for load, such as M_{22}^{L}.

The reason for this choice of disruptions is that both speed reduction and higher demand and backload will affect the solution enough to test the model's behavior regarding delay and postponing. Instances with IDs without superscript are non-disrupted. Disruption with an extra order (node) would only delay a vessel a couple of hours and is thus not tested in this report. It is though tested for one instance in Section 8.3, that is Å_{19}. This instance is similar to Å_{18}, only one extra delivery node is added. It is included because it is the largest instance without deck capacity as a binding resource that could be solved using the voyage-based solution method.

ID	**Depot**	**# Nodes**	**Size**	**# OSVs**
M_{22}	Mongstad	22	M	3
F_{24}	Florø	24	M	3
Å_{26}	Ågotnes	26	M	4

Table 8.1: Instances used to test the delay cost and the postpone cost in Section 8.2. #OSVs includes one spot vessel.

ID	Depot	# Nodes	Size	# OSVs
M_{12}	Mongstad	12	S	2
F_{24}	Florø	24	M	3
M_{42}	Mongstad	42	L	5

Table 8.2: Instances used to tune heuristic parameters in Section 8.2. #OSVs includes one spot vessel.

ID	Depot	# Nodes	Size	# OSVs
M_{10}	Mongstad	10	S	2
M_{12}	Mongstad	12	S	2
$Å_{14}$	Ågotnes	14	S	2
F_{16}	Florø	16	S	2
$Å_{18}$	Ågotnes	18	M	3
M_{22}	Mongstad	22	M	3
F_{24}	Florø	24	M	3
$Å_{26}$	Ågotnes	26	M	4
M_{42}	Mongstad	42	L	5
M_{54}	Mongstad	54	L	6

Table 8.3: Instances used to compare models in Section 8.3. #OSVs includes one spot vessel.

8.2 Penalty cost testing and heuristic parameter summary

This section gives a brief summary of the parameter values chosen for the input data and for the heuristic. Detailed results from the tests conducted are given in Appendix B. The parameters that are set in this section are as follows:

- C_v^D - the delay cost.
- C_i^R - the postpone cost.
- p_{max} - the maximum number of perturbations.
- s_{max} - the maximum number of shakings.
- n_{max} - the maximum number of times a solution is checked and rearranged.
- i_{max} - the number of initial solutions.
- The order in which the neighborhood move operators described in Section 7.4 are performed.

In this section, the instances described in Table 8.1 are used to test the delay and postpone costs, while the instances described in Table 8.2 are used to test the remaining parameters.

The instances used to test the postpone and delay costs were chosen since they service close to the maximum amount of installations serviced by the onshore depots at the time of writing. In addition, different depots are used to ensure that the costs are not tailored to a single depot. The authors have defined *acceptable* and *unacceptable* solutions when testing these two parameters, and the definitions can be found in Appendix B.1. Solutions that were deemed acceptable by the authors were found for high values of the two parameters, with postpone costs at NOK 200 000 for orders with normal priority and delay costs at NOK 50 000. For orders with higher priority the postpone costs are higher, details for this can found in Appendix B.1. These values are used for the remaining testing described in this report. It is important to note that these costs are fictional and only meant to illustrate that there are negative effects related to postponing orders and arriving at the depot after the planned schedule. They do not reflect any actual costs that occur in case of postponements or delays.

The parameters p_{max} and s_{max} that control the number of perturbations and shakings, respectively, were tuned simultaneously. This was done because these two parameters both affect how many times perturbation, and shaking and local search is run, and therefore affect each other and the heuristic's ability to find good solutions. They are calculated as $p_{max} = \lceil p \text{ x } N \rceil$ and $s_{max} = s \text{ x } N$, where $p \in \{0.5, 1, 2\}$ and $s \in \{5, 10, 20\}$ and N is the number of nodes in the problem. When the instances described in Table 8.2 were tested, the small instances M_{12} were not sensitive to the different values. To ensure a low run time, the smallest values for the parameters $p = 0.5$ and $s = 5$ were chosen. For the medium and large instances, high values for the parameters gave the best results. High parameter values means that the heuristic tries the different moves more times, and therefore has a better chance of finding good solutions. Thus, for the medium and large instances, $p = 2$ and $s = 20$ were chosen.

When testing the n_{max} parameter, the values obtained for p_{max} and s_{max} above were used. The n_{max} parameter controls the number of times a solution is checked and rearranged, and is calculated as $n_{max} = \lceil n \text{ x } (N/2) \rceil$, where $n \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7\}$ and N is the number of nodes in the problem. Again, the small instances were not sensitive to the different values tested, so $n = 0.1$ was chosen for these instances. For the remaining instances, high values again gave the best results, but as n reached 0.5 and higher, the differences in the objective values obtained were marginal. The run times of the medium sized instances did not increase much after this point was reached, so the highest value of $n = 0.7$ was chosen for these instances. The large instances however, had a much higher run time, and the value at the point where the good solutions were obtained, $n = 0.5$ was chosen.

The structure of the initial solution provided can affect the quality of the solution proposed by the heuristic. The construction heuristic has a large degree of randomness, and the best one of the i_{max} constructed solutions is given as input to the heuristic. If few solutions are constructed, the heuristic might not get a

good starting point, resulting in a poor solution. As with the previous parameters, the small instances were not sensitive to this parameter either. The lowest value $i_{max} = 5\ 000$ was therefore chosen. With increased problem size, there also exists a higher number of possible initial solutions, and the medium and large instances gave better results with high i_{max} values. For the medium instances, $i_{max} = 50\ 000$ was chosen, and $i_{max} = 100\ 000$ was chosen for the large instances.

In Section 7.4, the different neighborhood move operators were presented. The order in which these moves are performed can affect the final solution and the time used by the heuristic, based on what kind of operations they do and their computational time. The authors have only tested four different orders for the operators, since testing all possible orders would mean up to 9! possibilities. The list of the orders tested is given in Appendix B.2.4. Since the problem considers operational planning and disruption management, low run times are important, and the authors therefore chose to use the computationally cheapest moves first in all the tested orders. For example, when alternating between inter-route moves and intra-route moves, the computationally cheapest inter-route move would be followed by the cheapest intra-route move, and so on. Testing showed that Order 1, which performs the computationally cheapest moves first, gave good solutions for the tested instances. The order of the operators in this case is *insert*, *swap*, *shift*, *replace*, *2-opt*, *3-opt*, *cross*, *exchange*, and *sibling mover*. This order alternates between inter-route and intra-route moves. What this means, is that the heuristic alternates between moving nodes between two or more vessels, and rearranging nodes internally on one vessel. This results in nodes being tried in a large number of positions on several vessels.

In Table 8.4, the parameter values chosen for the different instances are summarized. These values are used in the small, medium, and large instances in Section 8.3. It is worth noting that this will not give the best possible result for all instances, but it would not be practical to choose different parameters for each single instance.

Instance type	p_{max}	s_{max}	n	init	Order
S	0.5	5	0.1	5 000	1
M	2	20	0.7	50 000	1
L	2	20	0.5	100 000	1

Table 8.4: Summary of the heuristic parameters chosen for the small, medium, and large instances. The Order decides the order in which the neighborhood operators are performed as described in Appendix B.2.4

8.3 Comparison of solution methods

In this section, the performance of the arc-flow formulation, the voyage-based solution approach, and the VNS heuristic are compared and discussed. First, the results from testing the problem instances where optimal solutions were found are presented. Then, the results from testing larger problem instances using the heuristic solution approach are shown and discussed. The problem instances used when comparing the three solution approaches, presented in Table 8.3, are also tested with disruptions. In Table 8.5, the objective values are presented for the voyage-based solution approach. The same values are found using the arc-flow model if optimality is proved. If the arc-flow model did not find optimality, the optimality gap is shown. The optimality gap is the gap between the best found solution and the best lower bound. For the heuristic, the average gap from the best found objective for ten runs is presented. The average gap is calculated as

$$Avg.\ gap = \frac{Average\ objective\ -\ Best\ found\ objective}{Best\ found\ objective}\ \text{x}\ 100\%$$

In Table 8.5, the *Best found objective* is the optimal solution found by the voyage based model, and in Table 8.6, it is the best found objective obtained with the heuristic in the ten runs for the instance. The time used by the voyage-based solution approach is the sum of the time used to generate voyages in Eclipse (Java) and the solution time used by Xpress.

Using commercial optimization software to solve the arc-flow formulation, optimal solutions are found for problem instances with up to 16 nodes, one regular OSV, and one spot vessel. In order to reduce the number of constraints, subtour eliminating constraints are added iteratively when the arc-flow model is run. Whenever an optimal solution, which may not be feasible, is found or the maximum time limit for an iteration is reached, a new iteration is initiated if the best current solution contains subtours. If a feasible solution is found when the time limit is reached, a new iteration is not initiated. The maximum time limit for an iteration is set to one hour. The results indicate that disruptions, such as speed reduction and increased order sizes, increase the computational time for solving the arc-flow formulation.

When the voyage-based solution approach is applied, the optimal solution is found for problem instances with up to 19 nodes, two regular OSVs and one spot vessel. I.e., optimal solutions are only proved for instances classified as *small* and some classified as *medium* in this report, and these instances are listed in Table 8.5. In general, the computational time increases as the number of possible voyages increases. Therefore, the computational time for the voyage-based solution approach is shorter when fewer voyages are feasible due to increased order sizes.

For some of the smallest problem instances, the voyage-based solution approach has a marginally lower computational time than the heuristic, seen in Table 8.5. However, since the computational time for the exact solution methods increases exponentially with the size of the problem instances, the heuristic is considerably faster for all realistic problem instances. As can be seen from Table 8.5, the average gap for the heuristic is 0.00% for all the instances where optimality is proven, i.e., it finds optimal solution in all ten runs. One can see that the average run time for the heuristic increases greatly for the Å_{18} instances compared to the other instances in Table 8.5. This is due to the classification of size, seen in Table 8.3. Medium and small sized instances have different parameters and this is reflected in the average run time. As explained in Section 8.1, the Å_{19} instance is included because this

		Arc-flow		Voyage	Heuristic	
	Opt. obj.	**Opt.**			**Avg.**	**AT[s]**
ID	**[NOK]**	**gap**	**Time[s]**	**Time[s]**	**gap**	
M_{10}	128 151	0.0%	0.1	0.13	0.0%	0.07
M_{10}^S	903 615	0.0%	7.2	0.06	0.0%	0.07
M_{10}^L	512 233	0.0%	3.0	0.03	0.0%	0.07
M_{12}	138 185	0.0%	0.4	0.51	0.0%	0.08
M_{12}^S	960 445	68.9%	3 602	0.38	0.0%	0.12
M_{12}^L	920 344	0.0%	709	0.04	0.0%	0.09
Å_{14}	179 089	0.0%	0.6	13.1	0.0%	0.12
Å_{14}^S	1 159 259	63.3%	18 004	13.0	0.0%	0.16
Å_{14}^L	1 084 404	38.7%	7308	0.56	0.0%	0.15
F_{16}	179 569	0.0%	0.4	529.3	0.0%	0.18
F_{16}^S	1 152 666	77.0%	14 403	553.4	0.0%	0.27
F_{16}^L	766 866	46.4%	104 643	101.3	0.0%	0.16
Å_{18}	319 480	69.6%	57 621	17 822	0.0%	9.83
Å_{18}^S	1 877 761	94.8%	64 826	9 727	0.0%	29.56
Å_{18}^L	321 544	68.6%	7 203	1 222	0.0%	11.18
Å_{19}	3 253 95	70.9%	144 039	103 147	0.0%	13.39

Table 8.5: Comparison of the three solution methods. For the arc-flow model the optimality gap (Opt. gap) is presented and for the heuristic, the average gap (Avg. gap) from the optimal objective value (Opt. obj.) found with the voyage based model is shown. For the heuristic, the average time (AT) in seconds over ten runs is presented.

is the largest instance without disruptions where the optimal solution is found by using the voyage-based solution approach. Since increased order sizes reduce the number of feasible voyages, the optimal solution was also found for the instance M_{22}^{L}. However, this result is not included in Table 8.3, since the optimal solution is not found for the corresponding instance without disruptions. The voyage-based solution approach could solve even larger instances if the loads were increased enough.

In Table 8.6, the results from testing the heuristic on the large and remaining medium instances are presented. Here, the heuristic is also run ten times to obtain the average values in the table.

From Table 8.5, it is clear that little time is used by the heuristic compared to the exact methods as the problem instances increase in size. As mentioned earlier, the run times of the voyage-based model and the arc-flow model increase exponentially. In an operational planning setting, the exact methods are not sufficient due to the run time. From the results presented in Table 8.5 and Table 8.6, it can be concluded that the heuristic finds optimal solutions for all instances where optimality is proven, and has a stable performance for both realistic sized problems and problems that are larger than what is encountered in real life today. Thus, one can see the advantage of using the heuristic for solving problems in a real life setting and use it as a decision tool on a day to day basis.

It is worth noting that as the problem instances increase in size, so does the run time of the heuristic, shown in Tables 8.5 and 8.6, due to the number of nodes and the parameters presented in Section 8.2. This is especially notable in the cases that have disruptions. The reason for this, is that the parameters used by the heuristic is dependent on the number of nodes. Among these, the most time consuming is the n_{max} parameter which controls the rearranging of the solution and reinserting of nodes. A non-disrupted case is not affected by this parameter since it by definition should not postpone any nodes. Hence for the large instances, a major increase in run time is seen for disrupted cases compared to the non-disrupted. This is

ID	Avg. obj. [NOK]	Avg. gap	AT[s]
M_{22}	303 055	0.00%	18.0
M_{22}^S	1 530 637	0.00%	27.1
M_{22}^L	996 884	0.00%	18.2
F_{24}	339 799	0.00%	23.0
F_{24}^S	1 680 057	0.00%	37.2
F_{24}^L	1 248 255	0.57%	58.9
Å_{26}	472 186	0.03%	39.9
Å_{26}^S	2 623 016	1.01%	142.3
Å_{26}^L	1 443 983	0.01%	43.8
M_{42}	582 094	0.02%	195.2
M_{42}^S	2 739 763	1.73%	394.0
M_{42}^L	1 946 880	2.73%	265.8
M_{54}	721 346	0.55%	367.5
M_{54}^S	3 216 599	0.70%	1 616
M_{54}^L	2 083 701	3.19%	1 686

Table 8.6: Results from testing the heuristic with medium and large instances. Average objective values (Avg. obj.), average gaps (Avg. gap) over the best found solution, and average run time (AT) are listed.

especially noticeable for the M_{54} cases. The parameters were not tuned for this instance, so despite that it is classified as a large instance, lowering for example the n_{max} value can result in lower run times, but could also result in poorer solutions. Even though the heuristic uses over 26 minutes to finish the largest instances with disruptions, the best final solution is usually found earlier. Figure 8.1 shows a plot of the objective values found throughout the run time of the heuristic for the instance M_{54}^L. Here, one can see that the last improvement for a run is usually found after approximately 800 seconds, with some exceptions. This means that

the operator of the model might stop the heuristic after this amount of time and still expect an acceptable solution. The figure shows that two solutions have an objective value approximately NOK 200 000 higher than the others. This is due to one extra postponement of a node in the two cases. The average objective and run time for the instance is illustrated with a red line in the figure.

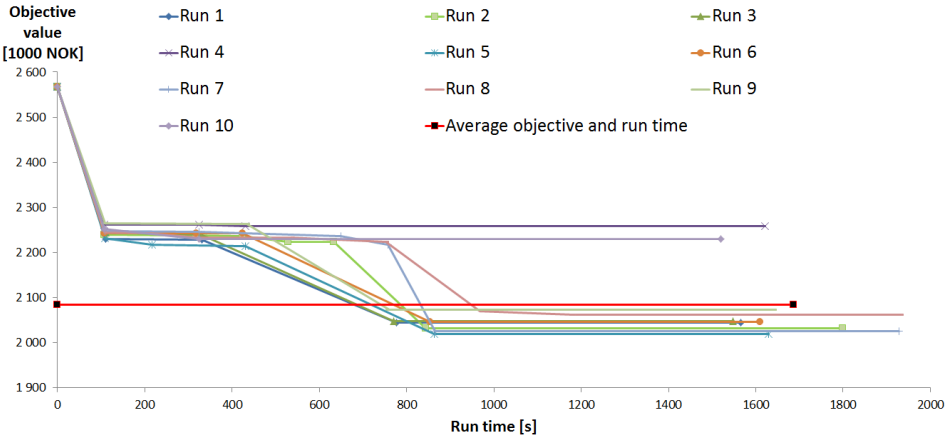

Figure 8.1: Run time and objective value for ten runs of M_{54}^{L} using the VNS heuristic.

Table 8.6 shows that $\mathring{\text{A}}_{26}^{S}$ is the smallest instance with an average gap higher than 1%. When the penalty costs, C_v^D and C_i^R, were decided, acceptable solutions were found for this instance using the parameters C_v^D=50 000 and C_i^R=200 000. The solutions had some delay and used the spot vessel and all three OSVs in the long-term fleet. However, when further inspecting all the solutions generated for the instance in this section, the heuristic finds solutions where the spot vessel is fully utilized for 72 hours and only one of the OSVs in the long-term fleet is used, which would be categorized as an unacceptable solution. The objective values for the two kinds of solutions are almost the same, sometimes even better for the unacceptable solutions. Since no orders are postponed, C_i^R does not affect the solution, but one might consider increasing C_v^D in order to utilize the OSV fleet

more efficient. One might wonder why it is less expensive to fully utilize the spot vessel instead of using the other available OSVs. One answer to this is the fact that the spot vessel has a 72 hour planning period and can sail longer distances than the OSVs in the long-term fleet, which can only sail for 40-45 hours in this instance before delay penalty costs are induced.

It is not only the run times that increase with the problem instance sizes. For the large disrupted cases, there is a noticeable gap between the best found objective and the average objective, whereas the gaps for the non-disrupted cases are below 1%. When the heuristic has to make a decision between postponing orders to ensure that the return time of the vessel is within the schedule and delaying vessels to ensure that all installations are serviced, more solutions exist than in the non-disrupted cases. Thus, there is also a higher risk that the heuristic gets stuck in local optima, which explains why there is a higher gap from the best found objective in these instances. The n_{max} parameter decides how many times the solution is to be rearranged and how many times reinsertion is tried, and one cannot guarantee that all possibilities are tried. Thus the same solution might not be obtained every time. The number of initial solutions generated can be increased since some randomization is used to generate initial solutions, but this will also affect the run time if i_{max} is too high. The high number of possible solutions that exists is one of the reasons why the exact solution methods are not able to solve the large instances.

As can be seen in Table 8.6 the highest average gaps for the heuristic are found for the large instances with disruption. When inspecting the solutions one can see that this is due to the different number of nodes that are postponed. Since the vessel capacity in these cases are fully utilized for both the OSVs in the regular fleet and the spot vessel, some nodes have to be postponed. Table 8.7 and Table 8.8 show a comparison of the objective values for the best and worst solutions found by the heuristic in the M_{42}^L and M_{42}^S instance, respectively. Here, *best* means the solution with lowest cost.

	Best solution	**Worst solution**	**Difference**
Objective value	1 895 169	2 101 525	206 356
Travel and service cost	1 295 169	1 301 525	6 356
Postpone cost	600 000	800 000	200 000
Delay cost	-	-	-

Table 8.7: Comparison of the best and worst objective values for the M_{42}^L instance. All numbers are in NOK and the difference is the value of the worst solution minus the best solution.

	Best solution	**Worst solution**	**Difference**
Objective function	2 693 127	2 790 399	97 272
Travel and service cost	1 893 127	1 904 399	11 272
Postpone cost	-	-	-
Delay cost	800 000	886 000	86 000

Table 8.8: Comparison of the best and worst objective values for the M_{42}^S instance. All numbers are in NOK and the difference is the value of the worst solution minus the best solution.

Table 8.7 and Table 8.8 show that the main difference between the objective values are the postpone and delay costs. As mentioned earlier, these are fictional costs and do not affect how the perturbation, shaking, and local search perform, since these procedures are only run on the solution after which nodes that are to be postponed is decided. When only considering the travel and service costs, and not the postpone and delay costs in the results for all ten solutions of the M_{42}^L instance, the average gap would only be 0.90%. This shows that the perturbation, shaking, and local search procedures perform very well even for the largest instances.

To increase the usability of the heuristic in practical settings, a tool generating graphical representations of the voyages found by the heuristic has been developed.

One illustration of a solution obtained with the heuristic for all instances described in this section, are given in Appendix D.

8.4 Analysis of suggested solutions

In the following sections, some of the solutions obtained by the presented heuristic are analysed and their applicability in real life situations is discussed. As shown in Section 8.3, the arc-flow model and the voyage-based solution approach have too high solution times to be practical for operational planning. Therefore, they are not considered in the following sections.

8.4.1 Detailed example on disruption handling

In this section, two of the small problem instances are presented to illustrate the use of the heuristic. First the M_{12}^L instance is discussed briefly and then the solutions found for the M_{10} and M_{10}^L instances are discussed in detail.

Even though an optimal solution is proved found, it is not necessarily an acceptable solution, as discussed in Section 8.2 and Appendix B.1. The reason for this is that the penalty cost parameters C_v^D and C_i^R are set to find acceptable solutions for medium sized instances. E.g. for the M_{12}^L instance four nodes are postponed and the spot vessel is not used in the optimal solution. This would be categorized as an unacceptable solution. Here, one might think that using the spot vessel would be a better solution instead of postponing 40% of the orders, but this is due to the trade-off between the cost of hiring a spot vessel and the cost of postponing nodes. The optimal solution for the M_{12}^L instance is illustrated in Figure D.6. In a real life setting, there will often be different preferences regarding how many postponed orders and how much delay that are acceptable. This can depend on the situation or the operator of the model. In these cases, the postpone and delay costs can be adjusted in order to obtain the preferred solutions.

Figure 8.2 shows the solution when the heuristic is used for operational planning

when there are no disruptions in the M_{10} instance. Figure 8.3, on the other hand, shows the solution for the disrupted case M_{10}^{L}. Here, the deliveries and backloads are tripled in size, which may occur if the installations have not been visited for some time due to e.g. bad weather. The solutions to both the instance with and the instance without disruptions are considered acceptable, and are presented to illustrate the model's capability of performing both operational planning and disruption management.

In the problems illustrated in Figure 8.2 and Figure 8.3, there are a total of ten pickup and delivery nodes. The figures however, only show five nodes corresponding to each installation, since there is one pickup and one delivery node located at the same installation in this example. The arcs that start and end in the same node depict visits to the delivery node and then the pickup node for the same installation.

In the disrupted example, no nodes were postponed, but Figure 8.3 shows that the solution suggests the use of a spot vessel. The cargoes on the arcs represent how much of the deck capacity that is in use between the different nodes. The vessels in this problem have a deck capacity of 1000 m^2. When OSV1 starts from the supply base (FMO), 975 m^2 of its deck capacity is used, making it impossible to visit the installation TRC which has a demand of 516 m^2. Thus, in this solution, a spot vessel must be chartered in order to service all the installations. In real life, one might consider dropping some demand or backload at each installation in order to service all the installations with one OSV, but this is not a possibility for the model presented in this report.

The voyage sailed by OSV1 in Figure 8.2 has a total cost of 128 151 NOK and takes 36 hours to complete. Since the greatest part of the distance sailed by an OSV is from the depot to the offshore installations and not between the installations, the time used by the spot vessel in Figure 8.3 is 21 hours even though it only visits one installation. In this case, OSV1 uses 32 hours and the total cost for using both OSV1 and the spot vessel is 512 233 NOK, which indicates how expensive it is to use a spot vessel.

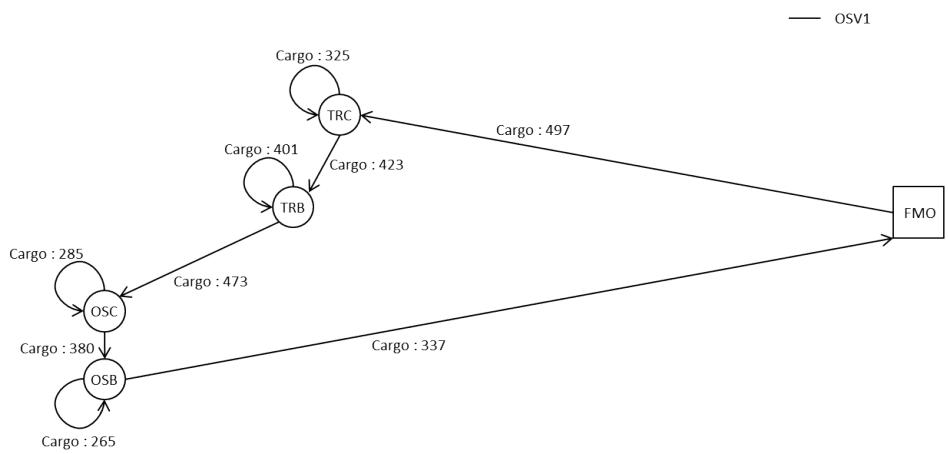

Figure 8.2: Optimal solution found for M_{10} with one OSV and no disruptions. The circles represent offshore installations, e.g. OSB is short for Oseberg B, the square is the supply depot (Mongstad), and the arcs are the voyages sailed by the vessels.

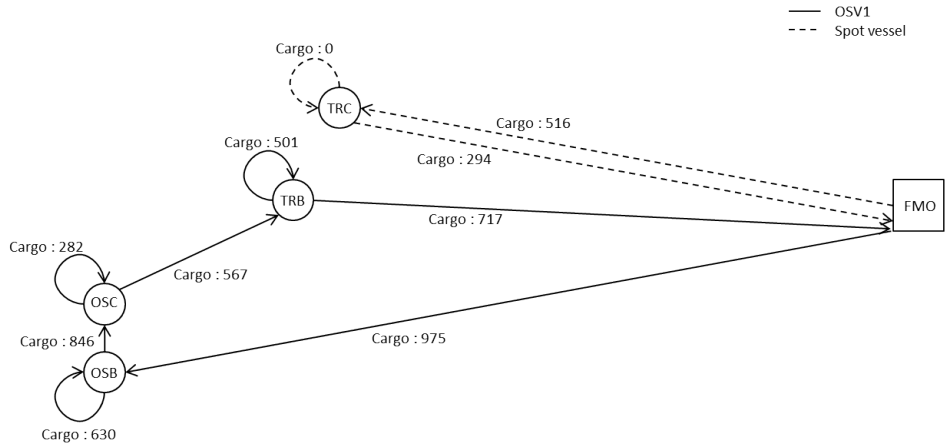

Figure 8.3: Optimal solution found for M_{10}^{L}. The spot vessel is used to handle the extra amount of cargo. The circles represent offshore installations, e.g. OSB is short for Oseberg B, the square is the supply depot (Mongstad), and the arcs are the voyages sailed by the vessels.

8.4.2 Comparison of a real life voyage and the suggested solution

In this section, a schedule which is carried out in real life is compared to the corresponding solution suggested by the heuristic. The comparison is made to illustrate that the heuristic provides reasonable results, and that it may help reducing the time spent on operational planning. The costs and the times needed to carry out the schedules are not fully comparable, since the presented model does not consider all issues encountered in real life.

Since the sailing speed may change during a voyage, the actual schedule is adjusted to be more comparable to the suggested solution, which assumes a sailing speed of ten knots during the whole voyage. The actual times needed to service the offshore installations are used when creating the problem instances solved by the heuristic. The voyages sailed in real life are illustrated in Figure 8.4, and the solution suggested by the heuristic is shown in Figure 8.5. Using the notation introduced in Section 8.1, the considered instance is denoted F_{26}, since the depot is Florø and the instance contains a total of 26 nodes including the depot.

When comparing the Figures 8.4 and 8.5, it is quite clear that the distances sailed by the vessels are shorter in the solution suggested by the heuristic than in the voyages which were actually sailed. The sailing times and costs for the two schedules are summarized in Table 8.9. The heuristic solved this problem in less than 30 seconds. An operational planner would most likely use significantly more time to schedule these voyages. The results indicate that savings with respect to both costs and sailing time can be made with the schedule suggested by the heuristic. The time saved can make the next voyages in the planning period depart earlier and possibly have time to visit more installations than planned. However, the presented model does not consider opening hours, crew changes, and visits by helicopters at the offshore installations. Since the operational planners must make such considerations in real life, schedules that are optimal according to the model

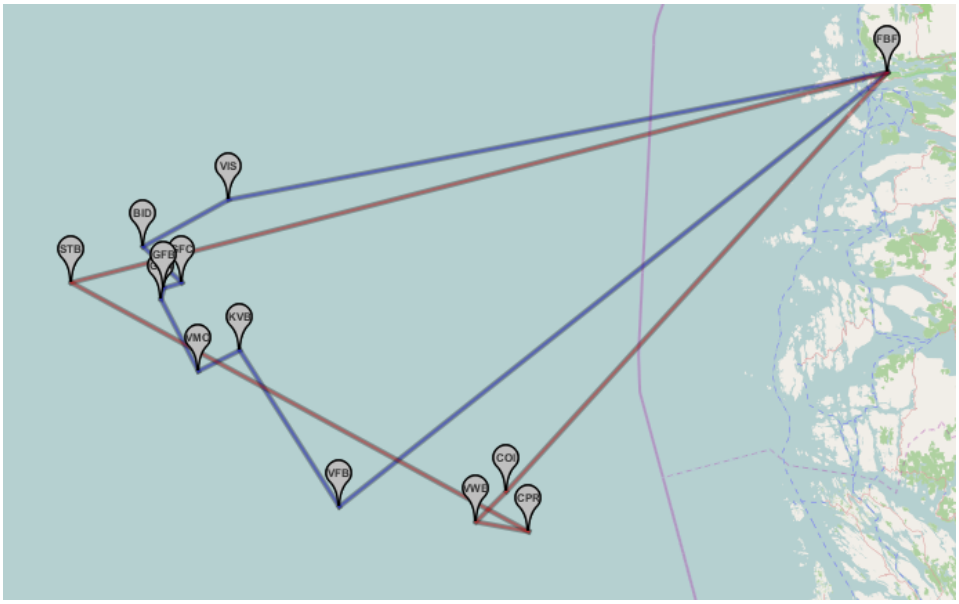

Figure 8.4: Actual voyages sailed in real life by two OSVs visiting 12 offshore installations.

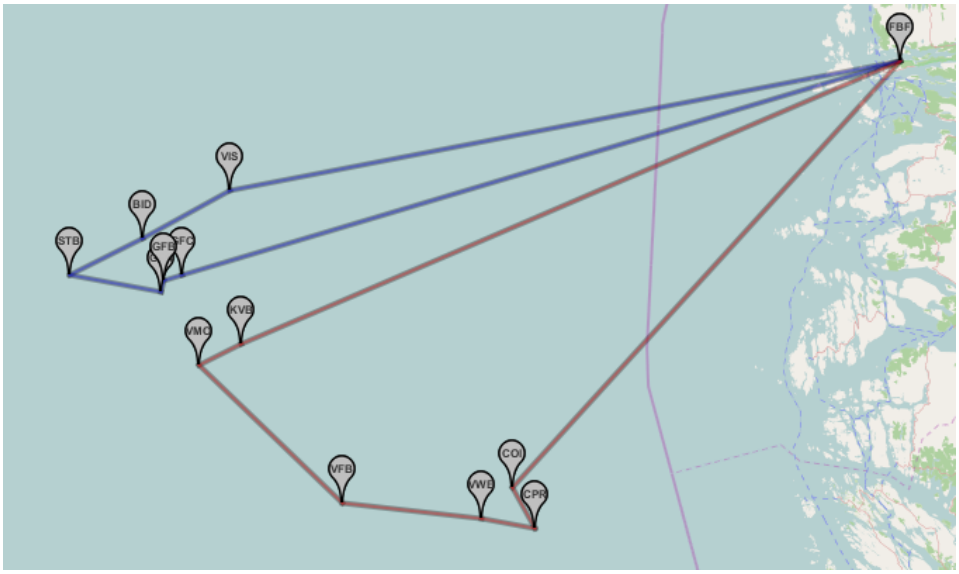

Figure 8.5: Suggested solution by the heuristic based on the real life case in Table 8.4.

presented in this report may not be possible to carry out in practice. Thus, when applying the heuristic for decision support in real life, the planners should adapt the solutions in order to handle the issues that the model does not consider. It is difficult to compare the results from the model with a real life case. However, Table 8.9 shows that the heuristic gives reasonable solutions given its limitations.

Actual schedule	**Time [hh:mm]**	**Cost [NOK]**
Voyage 1	39:35	142 919
Voyage 2	30:40	135 477
Total	70:15	278 396
Suggested schedule		
Voyage 1	33:07	134 169
Voyage 2	27:04	119 048
Total	60:11	253 217
Difference		
Voyage 1	06:28	8 750
Voyage 2	03:36	16 430
Total	10:04	25 179

Table 8.9: Comparison of actual and suggested schedule, with 12 offshore installations and two OSVs.

8.4.3 Planning with large numbers of installations

In this section, the operational planning of routes and schedules when a large number of installations is involved is discussed. As of today, maximum 13 offshore installations are serviced by a single supply depot. However, as mentioned, Statoil is planning to reduce the number of onshore depots. This implies that the remaining depots will service more installations than they do today. For instance, the Mongstad supply base will service as many as 26 installations. This will increase

the complexity of the planning of routes and schedules, and increases the motivation for using decision tools based on mathematical models in the operational planning.

In Figure 8.6, the solution of a problem instance with 42 nodes and increased order sizes for demand and backload is illustrated. The solution suggests using five OSVs (including one spot vessel) to service the installations, and postponing three of the 40 orders. The heuristic finds this solution within five minutes, whereas far more time would probably have been spent on manual planning due to the large amount of orders and vessels to be coordinated. When disruptions occur, it is important to take action quickly. A decision tool based on the presented heuristic can in short time provide the planners with a good solution, which can be further improved by their experience and knowledge. Thus, the time needed for planning routes can be greatly reduced, especially when a large number of installations is considered. In addition, better routes might be achieved, resulting in reduced costs in the offshore supply service.

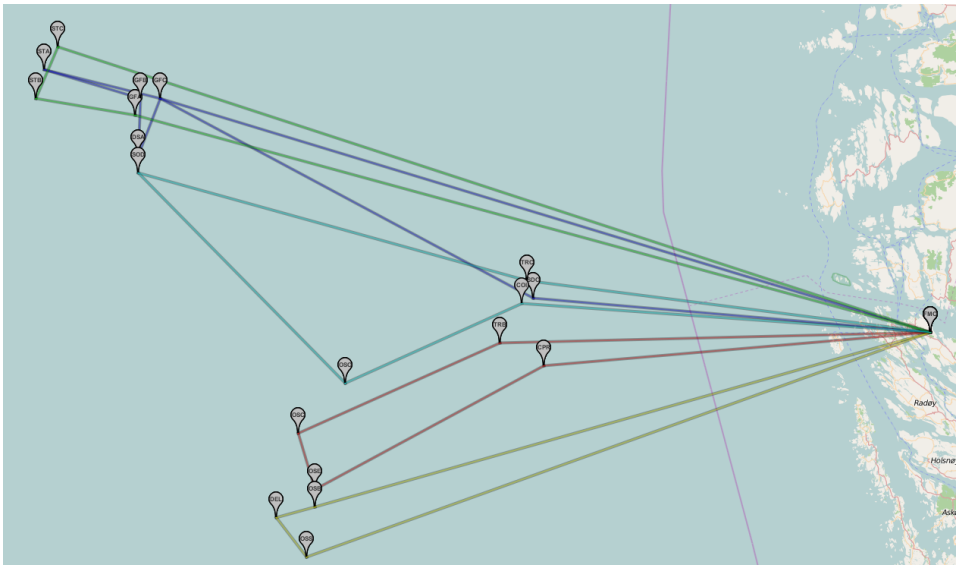

Figure 8.6: Suggested solution for the problem instance M_{42}^{L} illustrating the complexity of planning when a large number of installations is to be serviced. Three orders are postponed and a spot vessel is used.

Chapter 9

Concluding remarks

This report has discussed operational planning and disruption management in Statoil's offshore supply service. As of today, operational planning is based on the judgement and experience of professionals. The planning processes are time consuming and costly, and a decision tool based on mathematical models can provide significant savings.

The considered problem consists of offshore supply vessels (OSVs) delivering supplies to offshore installations from an onshore supply depot. In addition, the OSVs carry backload from the installations to the depot. The problem was modeled as a pickup and delivery problem with time limits for departure and arrival at the depot. The OSVs are either chartered on long-term contracts, or chartered from the spot market.

Both an arc-flow formulation and a voyage-based formulation representing the problem have been presented. Dynamic programming was applied for pregeneration of voyages in the voyage-based solution approach. Optimal solutions were found for instances with up to 19 nodes. For most problem instances of realistic size, optimal solutions could not be proven due to issues related to computational time and memory. Thus, the exact solution methods are not sufficient for decision

support in real life.

A variable neighborhood search (VNS) heuristic was developed in order to find solutions for all realistic problem instances. The heuristic was able to find optimal solution every time for all instances where optimality was proven by the exact solution methods. This shows that the heuristic produces high quality solutions, and it is reasonable to assume that the solutions for the instances of realistic sizes are optimal or close to optimal. The heuristic was tested for problem instances with up to 54 nodes. For all tested problem instances, the heuristic found solutions within reasonable time.

Using the proposed VNS heuristic for decision support, Statoil might reduce the time spent on operational planning and disruption management in their offshore supply service. In addition, better routes and schedules can be found, and significant savings may be achieved.

In future research, the modeled problem may be extended in several ways. Introducing time windows for opening hours and/or planned service from OSVs at offshore installations, could improve the service level for the offshore installations. Also, by using time windows, conflicting arrivals at the installations and the supply depot can be avoided. This could make the model more applicable in real life.

In addition, the presented model only considers cargo carried on deck, while an extension of the model could include capacity restrictions on bulk cargo as well. This would require the model to take the different kinds of bulk cargo into account, since the bulk products cannot be mixed and the vessels only contains a limited number of tanks. Also, some kinds of bulk cargo can only be carried by certain OSVs.

Chapter 10

References

Acuna-Agost, R., Michelon, P., Feillet, D., & Gueye, S. (2011). A mip-based local search method for the railway rescheduling problem. *Networks*, *57*(1), 69-86.

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I., & Laporte, G. (2007). Static pickup and delivery problems: a classification scheme and survey. *TOP*, *15*(1), 1-31.

Bratu, S., & Barnhart, C. (2006). Flight operations recovery: New approaches considering passenger recovery. *Journal of Scheduling*, *9*(3), 279-298.

Brouer, B. D., Dirksen, J., Pisinger, D., Plum, C. E., & Vaaben, B. (2013). The vessel schedule recovery problem (vsrp) – a mip model for handling disruptions in liner shipping. *European Journal of Operational Research*, *224*(2), 362 - 374.

Brønmo, G., Christiansen, M., Fagerholt, K., & Nygreen, B. (2007). A multi-start local search heuristic for ship scheduling—a computational study. *Computers & Operations Research*, *34*(3), 900-917.

Caseau, Y., & Laburthe, F. (1999, October). Heuristics for large constrained vehicle routing problems. *Journal of Heuristics*, *5*(3), 281–303.

Dell'Amico, M., Righini, G., & Salani, M. (2006). A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection.

Transportation Science, *40*(2), 235-247.

Dienst, D., Røpke, S., & Vaaben, B. (2012). Realistic models and computational results for disruption management in the airline industry. *Report from DTU Management Engineering*.

DNV GL. (2015). *A balancing act.* Retrieved 2015-04-14, from `http://www.dnv.com.cn/Binaries/Outlook%20for%20oil%20and%20gas%20industry%202015_tcm142-625443.pdf`

Eclipse.org. (2015). *Eclipse IDE for Java developers.* Retrieved 2015-05-25, from `http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/lunasr2`

Fagerholt, K., & Halvorsen-Weare, E. E. (2011). Optimization in offshore supply vessel planning. *Working paper*.

Fico.com. (2015). *FICO Xpress Optimization Suite.* Retrieved 2015-05-25, from `http://www.fico.com/en/products/fico-xpress-optimization-suite#overview`

Gribkovskaia, I., Halskau, O., Laporte, G., & Vlcek, M. (2007). General solutions to the single vehicle routing problem with pickups and deliveries. *European Journal of Operational Research*, *180*(2), 568-584.

Gribkovskaia, I., Laporte, G., & Shlopak, A. (2007). A tabu search heuristic for a routing problem arising in servicing of offshore oil and gas platforms. *Journal of the Operational Research Society*, *59*(11), 1449-1459.

Halvorsen-Weare, E. E., Fagerholt, K., Nonås, L. M., & Asbjørnslett, B. E. (2012). Optimal fleet composition and periodic routing of offshore supply vessels. *European Journal of Operational Research*, *223*(2), 508 - 517.

Huisman, D., Kroon, L. G., Lentink, R. M., & Vromans, M. J. C. M. (2005). Operations research in passenger railway transportation. *Statistica Neerlandica*, *59*(4), 467–497.

Karlsson, P. (2013). *A windy day offshore. psv volstad supplier and the drilling rig polar pioneer.* Retrieved 2014-05-23, from `http://www.flickr.com/photos/`

75210159@N06/9051998932

Kjeldsen, K. H. (2012). Routing and scheduling in liner shipping. *PhD*, 105 - 133.

Korsvik, J. E., Fagerholt, K., & Laporte, G. (2009). A tabu search heuristic for ship routing and scheduling. *Journal of the Operational Research Society*, *61*(4), 594-603.

Maisiuk, Y., & Gribkovskaia, I. (2014). Fleet sizing for offshore supply vessels with stochastic sailing and service times. *Procedia Computer Science*, *31*, 939 - 948.

Polat, O., Kalayci, C. B., Kulak, O., & Günther, H.-O. (2015). A perturbation based variable neighborhood search heuristic for solving the vehicle routing problem with simultaneous pickup and delivery with time limit. *European Journal of Operational Research*, *242*(2), 369-382.

Regjeringen. (2014). *Facts 2014 the norwegian petroleum sector.* Retrieved 2015-03-02, from https://www.regjeringen.no/globalassets/upload/oed/pdf_filer_2/faktaheftet/fakta2014og/fakta_2014_no_nett_.pdf

Scopus.com. (2015). *Scopus.* Retrieved 2015-03-02, from http://www.scopus.com

Shyshou, A., Gribkovskaia, I., Laporte, G., & Fagerholt, K. (2012). A large neighbourhood search heuristic for a periodic supply vessel planning problem arising in offshore oil and gas operations. *INFOR: Information Systems and Operational Research*, *50*(4), 195-204.

Sopot, E., & Gribkovskaia, I. (2014). Routing of supply vessels to with deliveries and pickups of multiple commodities. *Procedia Computer Science*, *31*(0), 910 - 917.

Statoil. (2014a). *Logistikkportalen.* Retrieved 2015-03-02, from http://www.logistikkportalen.no/forsyningskjeden/behov

Statoil. (2014b). *Procurement.* Retrieved 2015-03-02, from http://www.statoil.com/en/OurOperations/Procurement/Pages/default.aspx

Statoil. (2015a). *2014 statutory report.* Retrieved 2015-05-29, from http://www.statoil.com/no/InvestorCentre/AnnualReport/

AnnualReport2014/Documents/DownloadCentreFiles/01_KeyDownloads/Statutory_report_2014.pdf

Statoil. (2015b). *Logistikkportalen.* Retrieved 2015-06-01, from http://logistikkportalen.no/forsyningskjeden

Statoil. (2015c). *Logistikkportalen.* Retrieved 2015-03-02, from http://www.logistikkportalen.no/forsyningskjeden/behov/returbehov

Statoil. (2015d). *Logistikkportalen.* Retrieved 2015-03-02, from http://www.logistikkportalen.no/bibliotek/document_59

Steiger, M. (2015). *Jxmapviewer2.* Retrieved 2015-05-25, from https://github.com/msteiger/jxmapviewer2

Subramanian, A., Drummond, L., Bentes, C., Ochi, L., & Farias, R. (2010). A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, *37*(11), 1899 - 1911.

The Norwegian Oil Gas Association. (2015). *Homepage.* Retrieved 2015-03-04, from http://www.norskoljeoggass.no/en/

Appendices

Appendix A

Compact mathematical formulations

A.1 Arc-flow formulation

Sets		
$\mathcal{N}$	$\mathcal{N} = \{0, ..., n+m\}$	Set of all nodes (node 0 and node n+1 is depot pickup and delivery node, respectively)
$\mathcal{N}^{\mathcal{P}}$	$\mathcal{N}^{\mathcal{P}} = \{0, ..., n\}$	Set of all pickup nodes
$\mathcal{N}^{\mathcal{D}}$	$\mathcal{N}^{\mathcal{D}} = \{n+1, ..., n+m\}$	Set of all delivery nodes
$\mathcal{V}$	$\mathcal{V} = \{1, ..., k+1\}$	Set of all available OSVs (OSV k+1 is an OSV from the spot market)

Parameters

C^S_{vij}	$v \in \mathcal{V}, i \in \mathcal{N}, j \in \mathcal{N}$	Costs associated with sailing from node i to node j and servicing node j for OSV v
C^{TC}		Daily time charter cost for the OSV from the spot market
C^R_i	$i \in \mathcal{N}$	Penalty cost for postponing service to a node i
C^D_v	$v \in \mathcal{V} \setminus \{k+1\}$	Penalty cost per hour of delayed arrival at the onshore supply depot for an OSV v
D_i	$i \in \mathcal{N}^{\mathcal{D}}$	Cargo to be delivered measured in deck capacity at node i
P_i	$i \in \mathcal{N}^{\mathcal{P}}$	Cargo to be picked up measured in deck capacity at node i
Q_v	$v \in \mathcal{V}$	Deck capacity for OSV v
T_{vij}	$v \in \mathcal{V}, i \in \mathcal{N}, j \in \mathcal{N}$	Time needed for OSV v to sail from node i to node j and service node j
T^{MIN}_v	$v \in \mathcal{V}$	Planned next departure time from depot for OSV v
T^{MAX}_v	$v \in \mathcal{V}$	Planned arrival time at depot for the next voyage for OSV v

H		Length of a time charter period for the OSV from the spot market
τ		Maximum allowed delay for OSVs in the long-term fleet

Variables

x_{vij}	$v \in \mathcal{V}, i \in \mathcal{N}, j \in \mathcal{N}$	1 if OSV v sails from node i to node j on a voyage, otherwise 0
y_{vi}	$v \in \mathcal{V}, i \in \mathcal{N}$	1 if OSV v visits node i on a voyage, otherwise 0
u_i	$i \in \mathcal{N}$	1 if the order at node i is postponed, otherwise 0
l_{vij}	$v \in \mathcal{V}, i \in \mathcal{N}, j \in \mathcal{N}$	Load on board OSV v when sailing from node i to node j
t_v^D	$v \in \mathcal{V} \setminus \{k+1\}$	Total delay for OSV v (hours)
t^{TC}		Number of days that the OSV from the spot market is chartered

Objective

$$min \sum_{v \in \mathcal{V}} \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} C_{vij}^S x_{vij} + C^{TC} t^{TC} + \sum_{i \in \mathcal{N}} C_i^R u_i + \sum_{v \in \mathcal{V} \setminus \{k+1\}} C_v^D t_v^D \quad \text{(A.1)}$$

Constraints

$$\sum_{i \in \mathcal{N} \setminus \{0\}} x_{v0i} = 1 \qquad v \in \mathcal{V} \tag{A.2}$$

$$\sum_{i \in \mathcal{N} \setminus \{n+1\}} x_{vi(n+1)} = 1 \qquad v \in \mathcal{V} \tag{A.3}$$

$$\sum_{i \in \mathcal{N} \setminus \{0\}} x_{vi0} = 0 \qquad v \in \mathcal{V} \tag{A.4}$$

$$\sum_{j \in \mathcal{N}} x_{vji} - \sum_{j \in \mathcal{N}} x_{vij} = 0 \qquad v \in \mathcal{V}, i \in \mathcal{N} \setminus \{0, n+1\} \tag{A.5}$$

$$y_{vi} - \sum_{j \in \mathcal{N} \setminus \{i\}} x_{vij} = 0 \qquad v \in \mathcal{V}, i \in \mathcal{N} \tag{A.6}$$

$$\sum_{v \in \mathcal{V}} y_{vi} + u_i = 1 \qquad i \in \mathcal{N} \setminus \{0, n+1\} \tag{A.7}$$

$$l_{vij} \leq (Q_v - P_j) x_{vij} \qquad v \in \mathcal{V}, i \in \mathcal{N}, j \in \mathcal{N}^{\mathcal{P}} \tag{A.8}$$

$$l_{vij} \leq Q_v x_{vij} \qquad v \in \mathcal{V}, i \in \mathcal{N}, j \in \mathcal{N}^{\mathcal{D}} \tag{A.9}$$

$$l_{vij} \geq P_i x_{vij} \qquad v \in \mathcal{V}, i \in \mathcal{N}^{\mathcal{P}}, j \in \mathcal{N} \tag{A.10}$$

$$l_{vij} \geq D_j x_{vij} \qquad v \in \mathcal{V}, i \in \mathcal{N}, j \in \mathcal{N}^{\mathcal{D}} \tag{A.11}$$

$$l_{vij} \geq (P_i + D_j) x_{vij} \qquad v \in \mathcal{V}, i \in \mathcal{N}^{\mathcal{P}}, j \in \mathcal{N}^{\mathcal{D}} \tag{A.12}$$

$$\sum_{i \in \mathcal{N}} l_{vij} + P_j x_{vjh} - l_{vjh} + Q_v x_{vjh} \leq Q_v \qquad v \in \mathcal{V}, j \in \mathcal{N}^{\mathcal{P}}, h \in \mathcal{N} \tag{A.13}$$

$$\sum_{i \in \mathcal{N}} l_{vij} - D_j x_{vjh} - l_{vjh} + Q_v x_{vjh} \leq Q_v \qquad v \in \mathcal{V}, j \in \mathcal{N}^{\mathcal{D}}, h \in \mathcal{N} \tag{A.14}$$

$$\sum_{j \in \mathcal{N}^{\mathcal{D}}} D_j y_{vj} - l_{v0i} + Q_v x_{v0i} \leq Q_v \qquad v \in \mathcal{V}, i \in \mathcal{N} \tag{A.15}$$

$$l_{vi(n+1)} - \sum_{j \in \mathcal{N}^{\mathcal{P}}} P_j y_{vj} + Q_v x_{vi(n+1)} \leq Q_v \qquad v \in \mathcal{V}, i \in \mathcal{N} \tag{A.16}$$

$$t^{TC} \geq \left(\sum_{i \in N} \sum_{j \in N} T_{(k+1)ij} x_{(k+1)ij} \right) \frac{1}{H} \tag{A.17}$$

$$T_v^{MIN} + \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} T_{vij} x_{vij} - T_v^{MAX} \leq t_v^D \quad v \in \mathcal{V} \setminus \{k+1\} \tag{A.18}$$

$$t_v^D \leq \tau \quad v \in \mathcal{V} \setminus \{k+1\} \tag{A.19}$$

$$\sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S}} x_{vij} \leq |\mathcal{S}| - 1 \quad v \in \mathcal{V}, \mathcal{S} \subset \mathcal{N}, |\mathcal{S}| \geq 2 \tag{A.20}$$

$$x_{vij} \in \{0,1\} \quad v \in \mathcal{V}, i \in \mathcal{N}, j \in \mathcal{N} \tag{A.21}$$

$$y_{vi} \in \{0,1\} \quad v \in \mathcal{V}, i \in \mathcal{N} \tag{A.22}$$

$$u_i \in \{0,1\} \quad i \in \mathcal{N} \tag{A.23}$$

$$l_{vij} \geq 0 \quad v \in \mathcal{V}, i \in \mathcal{N}, j \in \mathcal{N} \tag{A.24}$$

$$t_v^D \geq 0 \quad v \in \mathcal{V} \setminus \{k+1\} \tag{A.25}$$

$$t^{TC} \in \mathbb{Z}^+ \tag{A.26}$$

A.2 Voyage-based formulation

Sets

$\mathcal{N}$	$\mathcal{N} = \{0, ..., n+m\}$	Set of all nodes
$\mathcal{V}$	$\mathcal{V} = \{1, ..., k+1\}$	Set of all available OSVs (OSV $m+1$ is the OSV from the spot market)
$\mathcal{R}_v$		Set of all possible voyages for OSV v

Parameters

C_{vr}^S	$v \in \mathcal{V}, r \in \mathcal{R}_v$	Costs associated with sailing voyage r for OSV v
C_i^R	$i \in \mathcal{N}$	Cost for postponing service to node i

C_v^D	$v \in \mathcal{V} \setminus \{k+1\}$	Cost per hour of delayed arrival at the onshore supply depot for OSV v
A_{ri}	$r \in \mathcal{R}_v, v \in \mathcal{V}, i \in \mathcal{N}$	1 if node i is visited on voyage r, otherwise 0
T_v^{MIN}	$v \in \mathcal{V}$	Planned next departure time from depot for OSV v
T_v^{MAX}	$v \in \mathcal{V}$	Planned last arrival time at depot for for the next voyage for OSV v
τ		Maximum allowed delay for OSVs in the long-term fleet

Variables

x_{vr}	$v \in \mathcal{V}, r \in \mathcal{R}_v$	1 if OSV v sails voyage r, otherwise 0
u_i	$i \in \mathcal{N}$	1 if the order at node i is postponed, otherwise 0
t_v^D	$v \in \mathcal{V} \setminus \{k+1\}$	Total delay for OSV v (hours)

Objective

$$min \sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{R}_v} C_{vr}^S x_{vr} + \sum_{i \in \mathcal{N}} C_i^R u_i + \sum_{v \in \mathcal{V} \setminus \{k+1\}} C^D t_v^D \tag{A.27}$$

Constraints

$$\sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{R}_v} A_{ri} x_{vr} + u_i = 1 \qquad i \in \mathcal{N} \setminus \{0, n+1\} \tag{A.28}$$

$$\sum_{r \in \mathcal{R}_v} x_{vr} \leq 1 \qquad v \in \mathcal{V} \tag{A.29}$$

$$T_v^{MIN} + \sum_{r \in \mathcal{R}_v} T_{vr}^S x_{vr} - T_v^{MAX} \leq t_v^D \qquad v \in \mathcal{V} \setminus \{k+1\} \tag{A.30}$$

$$x_{vr} \in \{0, 1\} \qquad v \in \mathcal{V}, r \in \mathcal{R}_v \tag{A.31}$$

$$u_i \in \{0, 1\} \qquad i \in \mathcal{N} \tag{A.32}$$

$$t_v^D \geq 0 \qquad v \in \mathcal{V} \setminus \{k+1\} \tag{A.33}$$

Appendix B

Parameter testing

This appendix contains the results from testing the effect of varying the postpone costs and delay costs, and the tuning of input parameters on the heuristic. A summary of the results is found in Section 8.2.

B.1 Analysis of penalty cost values

In this section, the focus is on finding the best possible values for:

- C_v^D - The cost per hour when OSV v arrives at the depot after schedule.
- C_i^R - The cost associated with postponing the service to node i.

The penalty cost C_v^D should be adjusted according to the negative effects of OSV v being delayed. This cost may vary depending on the weekday and the next planned voyage for the OSV. Planners applying the model in a real life case should adjust the parameter according to the current situation. For instance, if the spot charter rates are low, it may be preferable to charter an additional OSV from the spot market rather than allowing delays. To encourage such solutions, the planners can set a high value for C_v^D. The same might be done if the vessels have tight schedules after the current planning period and it is preferable to avoid delays.

In real life, the parameter value of C_i^R depends on the priority of the order i. If an order with low priority is postponed, it has less negative effects on operations at offshore installations than if an order with high priority is postponed. The money value for e.g. a small mechanical part needed on an installation might be low, but the importance might be high, and thus, the priority is correspondingly high. For the test instances considered in this section, 80% of the nodes have normal priority and 20% have high priority, with corresponding values for C_i^R. When using the models in real life cases there will be more than two possible priorities, and the cost of postponing an order should be adjusted according to these. For the Å_{19} instance, where one extra delivery node is introduced, the priority is set extra high since the node represents an important order.

In this section, the authors have chosen to test instances from three different onshore supply depots, namely Mongstad, Ågotnes, and Florø. This is done to ensure that the parameter values are not tailored for a single depot. Since the distances from one depot to the installations it services might be very different compared to another depot, the sailing cost will vary. I.e., the penalty costs will have different impact on the solutions for the three depots. The parameters are only tested on instances where disruptions are introduced. The reason for this is that in the non-disrupted cases, no delays or postponed orders occur.

How the parameter values fit the model is evaluated by whether the solutions to the modeled problem would be reasonable in real life scenarios or not. That is, the solutions utilize the available capacity in the long-term fleet rather than allowing significant delays, postponing several orders, and chartering additional vessels from the spot market. The solutions should keep the costs associated with sailing, servicing installations, and chartering OSVs low, without letting the vessels deviate too far from schedule.

To test the penalty cost parameters, six instances with disruptions are used. The two disruption types are given in Section 8.1, and the three sets of depots, nodes, and vessels are described in Table 8.1, giving a total of six instances.

It is challenging to find one value for each of the parameters to be used for all instances considered in this report. It is also difficult to evaluate whether a solution is acceptable or not. The criteria applied when evaluating solution quality in this report are as follows:

- The solution should not charter a spot vessel if the OSVs in the long-term fleet are not all fully utilized.
- Several nodes should be postponed or the OSVs in the long-term fleet should be significantly delayed before the spot vessel is to be used.
- Nodes should only be postponed if all vessels' deck capacity is fully utilized or visiting the node means that a vessel will be considerably delayed.

With respect to the criteria listed above, the solutions are either ranked as *acceptable* or *unacceptable*. Different values for C_v^D and C_i^R are tested. The value interval, the upper limit, and the lower limit are set by qualitative analysis. The values are then further tested and the solutions explored. In Table B.1, the results from the testing of C_v^D and C_i^R are shown. The colors indicate whether the solutions are acceptable, unacceptable, or not tested.

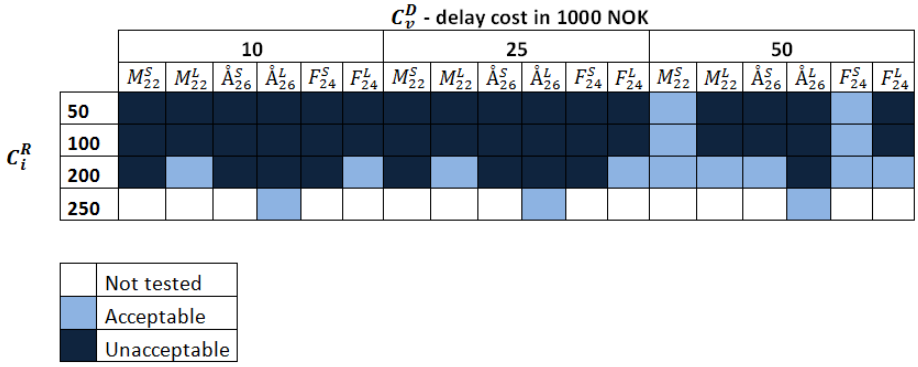

Table B.1: Ratings of solutions with different penalty values. C_i^R is the postpone cost in NOK 1000.

When evaluating the solutions, an average delay of more than 12 hours per OSV is considered as unacceptable. Most acceptable solutions are found for high delay costs (NOK 50 000), as seen in Table B.1. Low delay costs cause in most cases solutions where the vessels are more than 12 hours delayed, on average, and where the possibility of chartering an additional OSV from the spot market is not utilized. Acceptable solutions are obtained for all tested instances with C_v^D = NOK 50 000. Delay costs above C_v^D = NOK 50 000 are not tested for any of the instances, since there is little or no delay in the solutions.

As one can see in Table B.1, low postpone costs gives unacceptable solutions in most cases. When the postpone cost is low, the model suggests to postpone more nodes than what is reasonable. In general, high postpone costs (NOK 200 000-250 000) gives the best solutions. Considering that a high service level for the offshore installations is preferred, this is a reasonable result.

Testing shows that C_i^R = NOK 200 000 for orders with normal priority gives the best solutions. However, this value gives an unacceptable solution for instance $\mathring{\text{A}}_{26}^L$ for C_v^D = NOK 50 000 or lower. There is no delay, but several nodes are postponed. Increasing the postpone cost for orders with normal priority to C_i^R = NOK 250 000 gives an acceptable solution for the instance. Postpone costs above C_i^R = NOK 200 000 are not tested for any of the other instances since at most one order was suggested postponed in the solutions.

The test results show that different penalty cost values give different solution quality for the tested instances. When applied to real life cases, the user of the model should adjust the penalty costs to the current situation, that is, according to the depot, the number of nodes, current spot charter price etc. In the rest of this report, the following parameter values are used:

- C_v^D = 50 000 NOK/hour
- C_i^R = 200 000 NOK/order (Normal priority)
- C_i^R = 300 000 NOK/order (High priority)

- $C_i^R = 400\ 000$ NOK/order (Extra order, as in Å$_{19}$)

B.2 Heuristic parameter testing

In this section, the results from tuning the parameters used by the heuristic described in Chapter 7 are presented. Four major parameters affect the quality of the solutions :

- n_{max} - the maximum number of times the solution is checked and rearranged.
- p_{max} - the maximum number of perturbations that can be done without an improving solution.
- s_{max} - the maximum number of shakings that can be done without an improving solution.
- i_{max} - the number of initial solutions created.

Since both p_{max} and s_{max} affect how the perturbation, and shaking and local search perform, they are tuned simultaneously, and the results are presented in Appendix B.2.1. The parameter n_{max} defines the outer loop as described in Algorithm 7.1, and despite the fact that it determines how many times p_{max} and s_{max} are run, it does not in itself affect how the perturbation, and shaking and local search are performed. For this reason, n_{max} is tuned after p_{max} and s_{max}, and the results are presented in Appendix B.2.2.

In addition to these three parameters, the number of initial solutions generated can also affect the quality of the solutions obtained. The results from testing the number of initial solutions are showed in Appendix B.2.3. Furthermore, the order of the neighborhood operators introduced in Section 7.4 can affect the solutions and the speed at which they are achieved, and the results from the order testing are presented in Appendix B.2.4.

In order to make the parameters applicable for both regular operational planning and disruption management, the instances used in the tuning are both non-disrupted and disrupted cases. In the following, the intervals for which the parameters are tested are found through qualitative analysis.

B.2.1 Perturbation and shaking parameters

The tests conducted with the heuristic in the remainder of this chapter are presented as an average over ten runs of the heuristic. Instead of showing the average objective value, the gap from the best found objective for the instance is shown, calculated as

$$Avg.\ gap = \frac{Average\ objective - Best\ found\ objective}{Best\ found\ objective} \text{ x } 100\%$$

In the following tables, AT indicates the average time in seconds it took to run the instances. The green cells mark the solutions with the parameter values the authors deemed the best for each instance. The M_{12} cases are not presented in this section, since tests showed that they were not sensitive to the parameters. Optimal solutions for these instances were obtained every time. Thus, for the small instances in this report, the lowest setting of $p_{max} = 0.5$ x N and $s_{max} = 5$ x N was chosen, which results in the lowest run time. Here, N is the number of nodes in the instance.

The results from the Florø (F_{24}) instances are shown in Table B.2. Since the average gap over the best found objective decreases when the values of p_{max} and s_{max} increase, the stability of the heuristic improves. This is because increased parameter values result in the heuristic trying the moves a larger number of times, and therefore has a better chance of finding good solutions. The only instance that has a gap for all the tested values is F_{24}^{L}, and for this instance, the average gap from the best known objective is lowest when $p_{max} = 2$ x N and $s_{max} = 20$ x N. For this reason, these parameter values are chosen for the medium sized instances.

The results from the Mongstad (M_{42}) instances are shown in Table B.3. As with

		$p_{max} = 0.5$ x N		$p_{max} = 1$ x N		$p_{max} = 2$ x N	
ID	**s**	**Avg. gap**	**AT[s]**	**Avg. gap**	**AT[s]**	**Avg. gap**	**AT[s]**
	5	0.03%	2.9	0.05%	4.7	0.00%	7.3
F_{24}	**10**	0.03%	4.4	0.01%	8.1	0.00%	13.1
	20	0.00%	8.0	0.00%	13.8	0.00%	23.4
	5	0.00%	4.6	0.00%	6.9	0.00%	11.4
F_{24}^{S}	**10**	0.00%	6.9	0.00%	11.6	0.00%	21.5
	20	0.00%	12.5	0.00%	20.1	0.00%	40.8
	5	0.59%	3.6	0.41%	6.4	0.50%	11.4
F_{24}^{L}	**10**	0.60%	6.0	0.51%	10.5	0.37%	21.8
	20	0.41%	11.2	0.42%	22.1	0.32%	44.6

Table B.2: Testing of p_{max} and $s_{max} = s$ x N values on the Florø instances with N = 24 nodes. Average gap (Avg. gap) over best found objective for the instances are shown, along with average run time in seconds (AT).

the Florø instances in Table B.2, the average gap over the best found objective decreases as the values of p_{max} and s_{max} increase. It is worth noting that the average gaps are larger for the M_{42} instances than for the F_{24} instances which had minor gaps, and the M_{12} instances which had no gaps. One reason for this is that the M_{42} instance has more nodes, and therefore more possible solutions to check than the smaller instances have. The M_{42} instance has a depot that service more installations than any of Statoil's onshore depots do at the time this report is written.

It is also clear that the non-disrupted case has a much more stable and better performance than the disrupted cases, illustrated by the lower average gaps from the best found objective in Table B.3. This is mainly due to the fact that the routes in the non-disrupted cases are not delayed, does not have any postponed nodes,

		$p_{max} = 0.5$ x N		$p_{max} = 1$ x N		$p_{max} = 2$ x N	
ID	**s**	**Avg. gap**	**AT[s]**	**Avg. gap**	**AT[s]**	**Avg. gap**	**AT[s]**
	5	0.36%	20.4	0.27%	33.1	0.27%	66.4
M_{42}	**10**	0.08%	36.3	0.06%	61.0	0.06%	112.5
	20	0.03%	67.7	0.04%	100.7	0.01%	232.5
	5	8.37%	28.5	7.02%	55.4	3.21%	126.1
M_{42}^S	**10**	5.02%	55.8	2.98%	113.3	2.54%	254.7
	20	4.69%	108.6	2.22%	199.1	1.96%	480.8
	5	5.71%	37.4	4.51%	82.0	4.12%	137.7
M_{42}^L	**10**	4.64%	74.4	3.93%	138.9	3.78%	291.3
	20	4.16%	148.8	3.60%	269.2	3.48%	526.5

Table B.3: Testing of p_{max} and $s_{max} = s$ x N values with Mongstad instances where N = 42 nodes. Average gap (Avg. gap) over best found objective for the instances are shown, along with average run time in seconds (AT).

and does not use the spot vessel. Thus, it is fewer possible solutions that can be explored by the heuristic, leading it to find the same good solutions more often. On the contrary, the disrupted cases have to deal with postponements, delays, and the use of a spot vessel. This leads to a large amount of different solutions being checked by the heuristic. For example, it must be considered whether a vessel should be delayed by a given number of hours, or whether a node should be postponed to ensure that the return time to the depot according to schedule is adhered. This is controlled by the delay and postpone penalty costs, which are set to guide the heuristic towards good solutions, but could be changed by an operator using the model.

Better results might be achieved with higher values of p_{max} and s_{max}, but this would also increase the run time, which should be low in operational planning and

disruption management. The lowest average gaps are found when p_{max} = 2 x N and s_{max} = 20 x N for all three instances, and are therefore the values chosen for the large sized instances.

B.2.2 Rearrange parameter

The n_{max} parameter represents the maximum number of times the solution is checked and rearranged. When the parameter is tuned, the best values found for p_{max} and s_{max} in the previous section are used. The n_{max} value is calculated as follows

$$n_{max} = \lceil n \text{ x } (N/2) \rceil$$

where n is in the interval [0.1, 0.7] and N is the number of nodes in the problem instance. The n_{max} values and the corresponding average gaps (Avg. gap) and average times (AT), explained in Appendix B.2.1, are given in Table B.4 and Table B.5 for the disrupted cases of F_{24} and M_{42} respectively. The non-disrupted cases are not used to tune n_{max} since this parameter is set equal to three in Algorithm 7.3 for these cases. Thus, the initial value of n_{max} does not affect the performance or the run time for the non-disrupted cases.

The M_{12} cases are not presented in this section, since they were not sensitive to this parameter. Thus, the value resulting in the lowest run time, $n_{max} = \lceil 0.1 \text{ x } (N/2) \rceil$, is chosen for the small instances in this report.

For the F_{24}^S instance results shown in Table B.4, there is no change in the average gaps as the n_{max} value increases, only in the run times. The results for the F_{24}^L instance however, show that as the n_{max} value increases, the average gaps decrease. Despite increased run times as the n_{max} value increases, the value $n_{max} = 9$ provides the best results. Thus, the value $n = 0.7$ is chosen for the medium instances.

The results from the disrupted M_{42} instances are shown in Table B.5. As with the F_{24} cases, the average gaps over the best found objective decrease as the n_{max}

		F_{24}^S		F_{24}^L	
n	n_{max}	**Avg. gap**	**AT[s]**	**Avg. gap**	**AT[s]**
0.1	**2**	0.00%	27.1	0.64%	16.9
0.2	**3**	0.00%	39.7	0.54%	25.2
0.3	**4**	0.00%	38.5	0.52%	38.6
0.4	**5**	0.00%	37.8	0.45%	44.8
0.5	**6**	0.00%	39.7	0.26%	62.6
0.6	**8**	0.00%	38.6	0.18%	63.6
0.7	**9**	0.00%	39.9	0.08%	83.4

Table B.4: Testing of $n_{max} = \lceil n \text{ x } (N/2) \rceil$ values on the Florø cases where N = 24 nodes. Average gap (Avg. gap) over best found objective for the instances are shown, along with average run time in seconds (AT).

		M_{42}^S		M_{42}^L	
n	n_{max}	**Avg. gap**	**AT[s]**	**Avg. gap**	**AT[s]**
0.1	**3**	4.80%	143.6	11.81%	107.4
0.2	**5**	4.12%	287.1	8.59%	261.9
0.3	**7**	2.09%	368.0	2.88%	476.4
0.4	**9**	0.80%	487.7	2.80%	516.7
0.5	**11**	0.73%	538.1	2.76%	528.0
0.6	**13**	0.70%	665.7	2.67%	628.1
0.7	**15**	0.78%	654.6	2.55%	769.0

Table B.5: Testing of $n_{max} = \lceil n \text{ x } (N/2) \rceil$ values on the Mongstad instances where N = 42 nodes. Average gap (Avg. gap) over best found objective for the instances are shown, along with average run time in seconds (AT).

value increases, indicating that the stability increases with the parameter value. The M_{42}^S instance has an average gap below 0.8% for n_{max} values of nine and higher, and thus all of these values give stable solutions. It is worth noting that the M_{42}^L instance has a very high gap when the n_{max} value is low. This is because several attempts on dropping and reinserting of nodes are needed before a good solution is obtained for this instance. When the n_{max} value is seven or higher, the solutions obtained for M_{42}^L are more stable. A n_{max} value of 11 gives a stable solution for both M_{42}^S and M_{42}^L, while keeping the run time at a reasonable level. This leads to a $n = 0.5$ being chosen for the large instances.

B.2.3 Number of initial solutions

As explained in Section 7.2 the initial solution in Algorithm 7.2 is found by comparing i_{max} number of generated solutions. In this section the results from testing the heuristic for a number of different i_{max} values are presented. The five values that are tested for i_{max} are in the interval [5000, 100 000], and the heuristic is run ten times for every considered value. One small, medium and large instance with and without disruptions are tested, and these are shown in Table 8.2. To find the best possible i_{max} for every instance, the average gap and the average time, explained in Appendix B.2.1, are found.

When the testing of the M_{12} cases was conducted, it showed that the average gap was equal zero for every case of i_{max}. This was also the case for F_{24} and F_{24}^S. For F_{24}^L the maximum average gap was 0.15%. The authors finds this gap negligible, and M_{12} and F_{24} is therefore not further discussed in this section. For the small instances i_{max} is set to 5 000, and 50 000 for medium instances based on the average run time.

As in the previous sections, the results for the M_{42} instance with no disruption, are good for every value of i_{max}. When it comes to M_{42} with triple load and reduced speed, the average gap goes from 4% to 2.2% for $i_{max} = 5\,000$ and $i_{max} = 100\,000$,

respectively. This is the case for both disrupted instances and is caused by poor initial solutions when i_{max} is low. The reason why the heuristic is more unstable for the disrupted cases is discussed in Appendix B.2.1. On a general basis, the average gap decreased when i_{max} increased. Since Algorithm 7.2 uses some randomization, it is obvious that higher i_{max} will on average result in better initial solutions than lower values. When it comes to the total run time, this was not affected greatly by the number of initial solutions that was generated. In the M_{42}^S and M_{42}^L cases, the generation of the initial solutions took around five seconds, and is thus only a fraction of the total run time (250-600 seconds). Based on the results presented above, i_{max} is set to 100 000 for large instances.

B.2.4 Move operator order

In the Algorithms 7.4 and 7.5, the shaking and local search for the VNS are described. In both algorithms, eight neighborhood operators are used. If the initial solution provided has any delays, a ninth operator, the sibling mover operator, is also used. In this section, the results from testing different orders of the operators in which they were performed are presented. Since the operators being performed first, are performed more often, the authors chose the computationally cheapest operators to be performed first if possible. For example, when alternate inter-route and intra-route moves are tested, the cheapest inter-route move is followed by the cheapest intra-route move and so forth. The same orders are used for both the shaking and local search in the Algorithms 7.4 and 7.5. Four different orders were tested, and are subsequently described below.

- **Order 1**(computationally cheapest first) : *insert, swap, shift, replace, 2-opt, 3-opt, cross, exchange, sibling mover.*
- **Order 2**(first intra-route moves, then inter-route moves) : *insert, swap, 2-opt, 3-opt, shift, replace, cross, exchange, sibling mover.*

- **Order 3**(first inter-route moves, then intra-route moves) : *shift, replace, cross, exchange, sibling mover, insert, swap, 2-opt, 3-opt.*
- **Order 4**(Alternate between inter-route and intra-route moves) : *shift, insert, replace, swap, cross, 2-opt, exchange, 3-opt, sibling mover.*

The testing was limited to these four combinations, since all possible combinations would mean testing up to 9! different orders, based on the problem instance. The first order is chosen because performing all the cheapest operators first might result in low run times. Order 3 is chosen because it first tries to move nodes between vessels, and then move nodes internally on each vessel. Order 2 is chosen to see if doing the opposite of Order 3 has any effects. Finally, Order 4 is chosen to test the effect of alternately moving nodes between vessels and internally on vessels.

As in the testing of p_{max}, s_{max}, n_{max} and i_{max}, the M_{12} instances were not sensitive to the different order of neighborhood operators, and are therefore not discussed further.

The tests showed that the non-disrupted cases were very little sensitive to the different orders, both with respect to run time and average gap over best found objective. It also became clear that as the problem instance size increased, so did the average gap, and naturally the run time since this is affected by the number of nodes. The highest gaps at up to 5% were found when testing the M_{42}^S and M_{42}^L instances. However, the heuristic's performance was stable for most of the orders. The order that was deemed by the authors to give the best results for the F_{24} and M_{42} instances (with and without disruptions), was Order 1. Order 1 is therefore chosen as the standard operator order for all instances.

Order 1 alternates between inter-route and intra-route moves. What this means, is that the heuristic alternates between moving nodes between two or more vessel, and rearranging nodes internally on one vessel. This results in nodes being tried in a large number of positions on several vessels.

Appendix C

Code and test instances

All programmed code and test instances are provided as a zipped archive containing:

1. The arc-flow folder which contains
    - Xpress-Mosel file for the arc-flow model.
    - Data files for the problem instances in Table 8.5.
2. The voyage-based folder which contains
    - Xpress-Mosel file for the voyage-based model.
    - Data files to use in Xpress-Mosel for the problem instances in Table 8.5 generated in Java.
3. The heuristic folder which contains
    - A Java workspace that contains code and data files for the voyage generation and the VNS heuristic.
    - Results from ten runs of the VNS heuristic for each problem instance in Table 8.5 and Table 8.6.
    - Text file with results explanation.

Appendix D

Solution figures

In this appendix, one solution for each instance, presented in Table 8.5 and Table 8.6, is illustrated using the map drawer that the authors have implemented in Java. The map drawer is based on the JXMapViewer2 project by Steiger (2015), and uses the results from the heuristic. The structure of the figure label is as follows:

- ID: [Instance ID]
- Spot: ["Yes" if spot vessel is used, "No" otherwise]
- Delay: [The total number of hours the vessels in the solution are delayed]
- Postponed: [Total number of nodes that are postponed]
- Objective: [The objective value for the solution]
- Run time: [The run time for the heuristic]

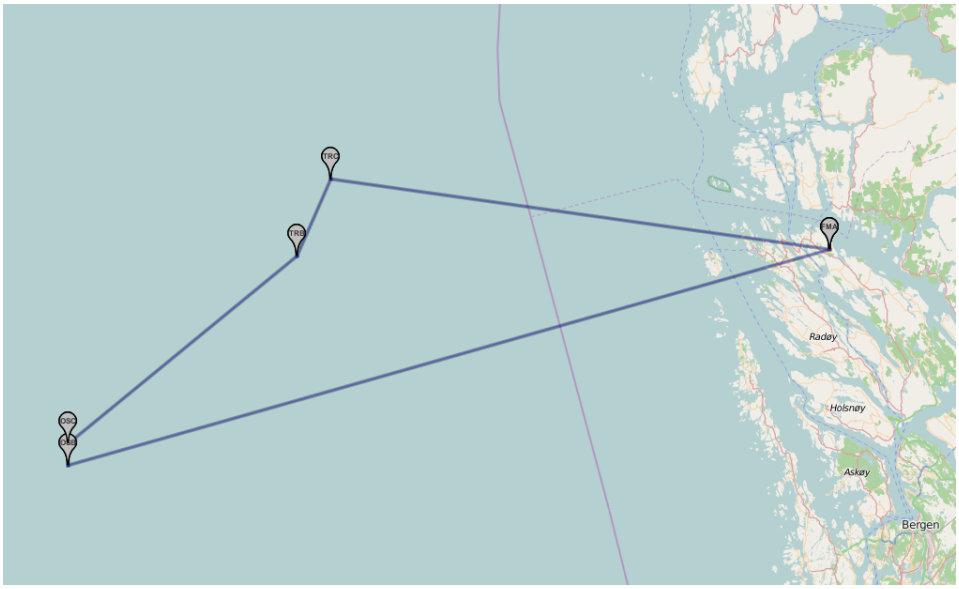

Figure D.1: ID: M_{10}, Spot: No , Delay: 0 hours, Postponed: 0 nodes, Objective: NOK 128 151, Run time: 0.25 seconds.

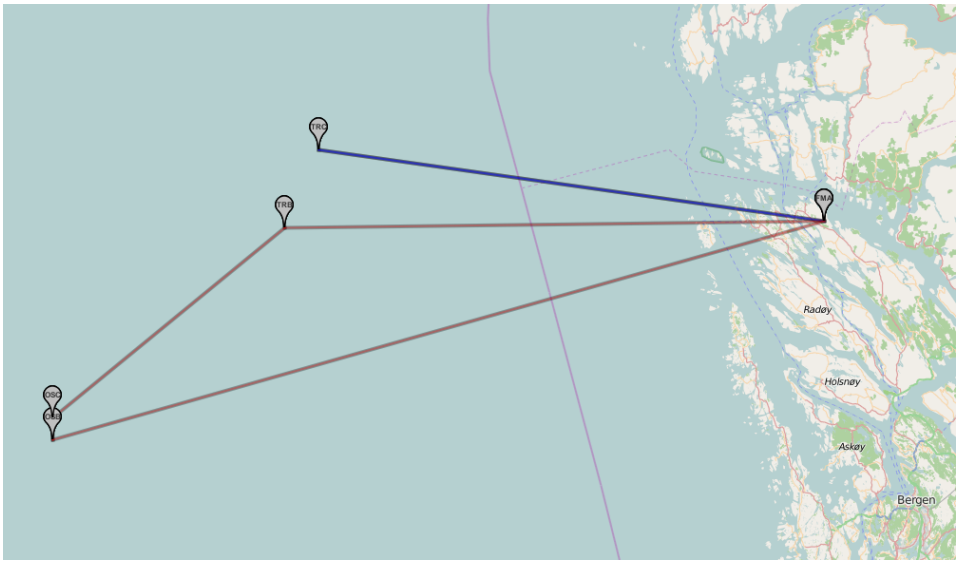

Figure D.2: ID: M_{10}^{S}, Spot: Yes , Delay: 0 hours, Postponed: 0 nodes, Objective: NOK 903 615, Run time: 0.12 seconds.

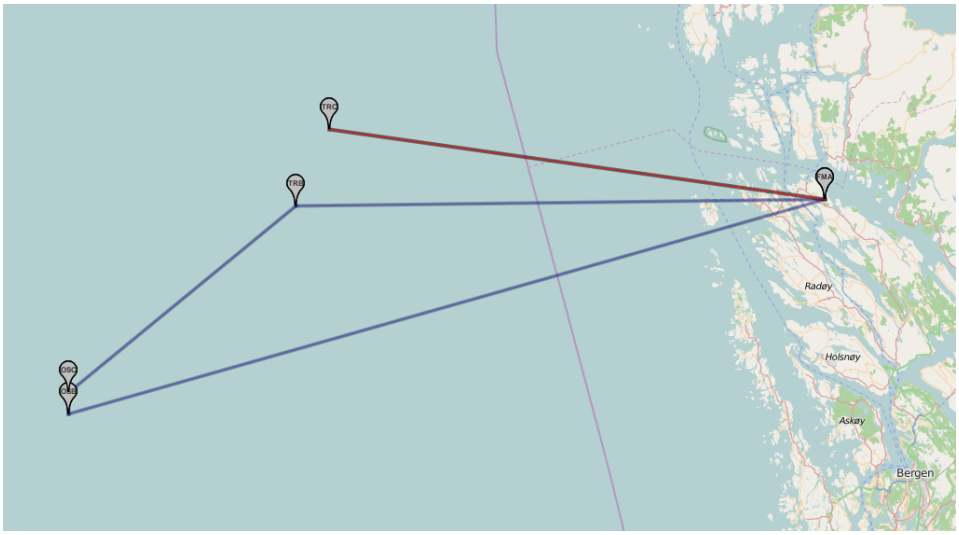

Figure D.3: ID: M_{10}^{L}, Spot: Yes, Delay: 0 hours, Postponed: 0 nodes, Objective: NOK 512 233, Run time: 0.089 seconds.

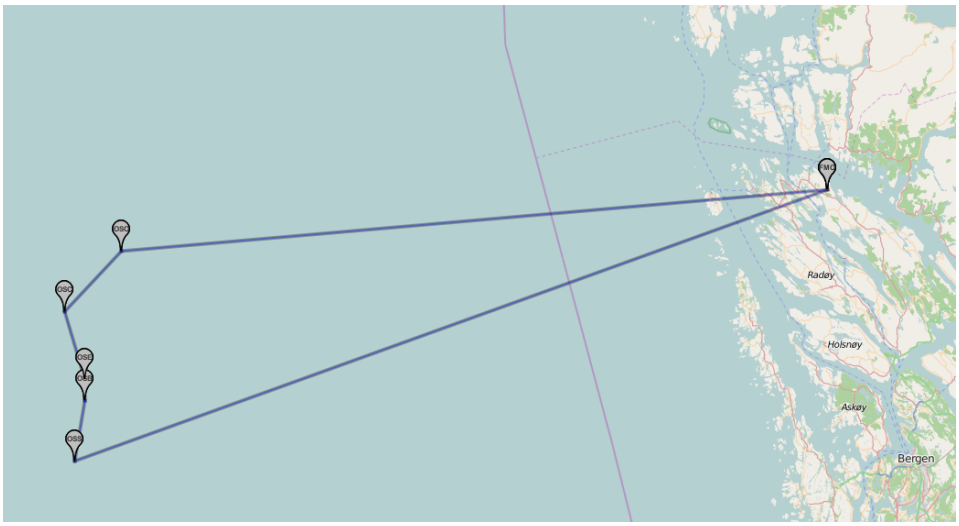

Figure D.4: ID: M_{12}, Spot: No, Delay: 0 hours, Postponed: 0 nodes, Objective: NOK 138 185, Run time: 0.13 seconds.

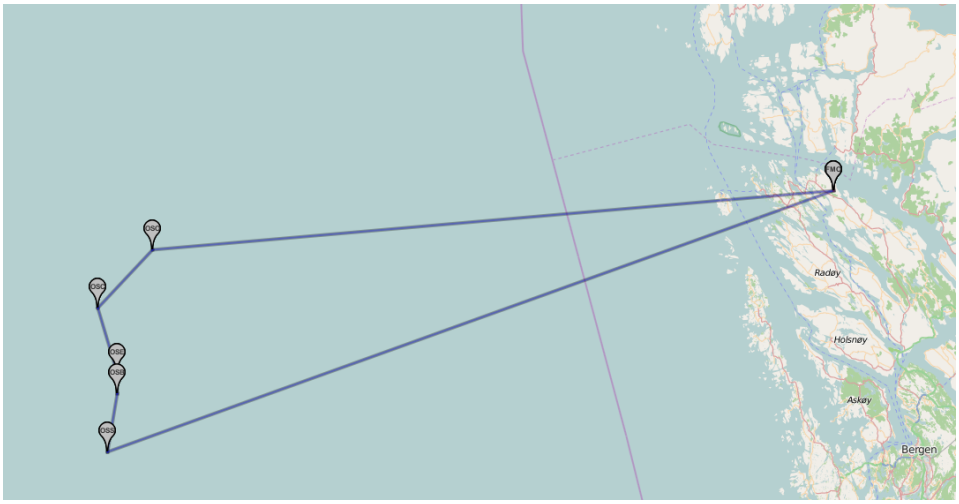

Figure D.5: ID: M_{12}^{S}, Spot: No, Delay: 15 hours, Postponed: 0 nodes, Objective: NOK 960 445, Run time: 0.17 seconds.

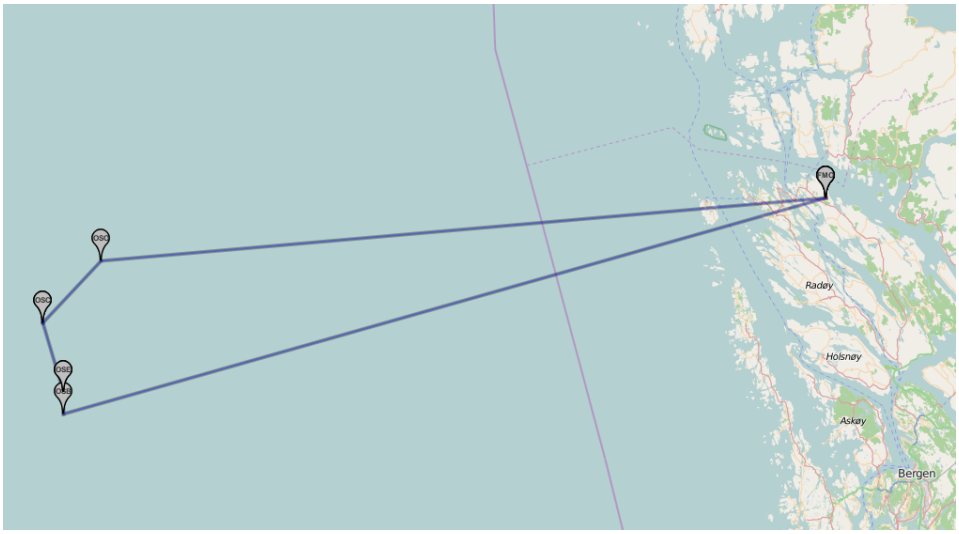

Figure D.6: ID: M_{12}^{L}, Spot: No, Delay: 0 hours, Postponed: 4 nodes, Objective: NOK 920 344, Run time: 0.13 seconds.

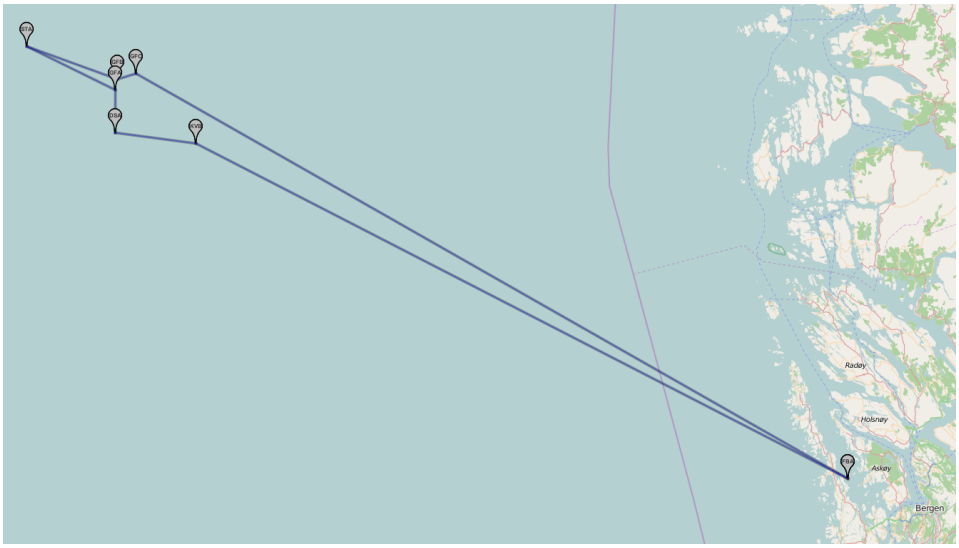

Figure D.7: ID: Å_{14}, Spot: No, Delay: 0 hours, Postponed: 0 nodes, Objective: NOK 179 089, Run time: 0.26 seconds.

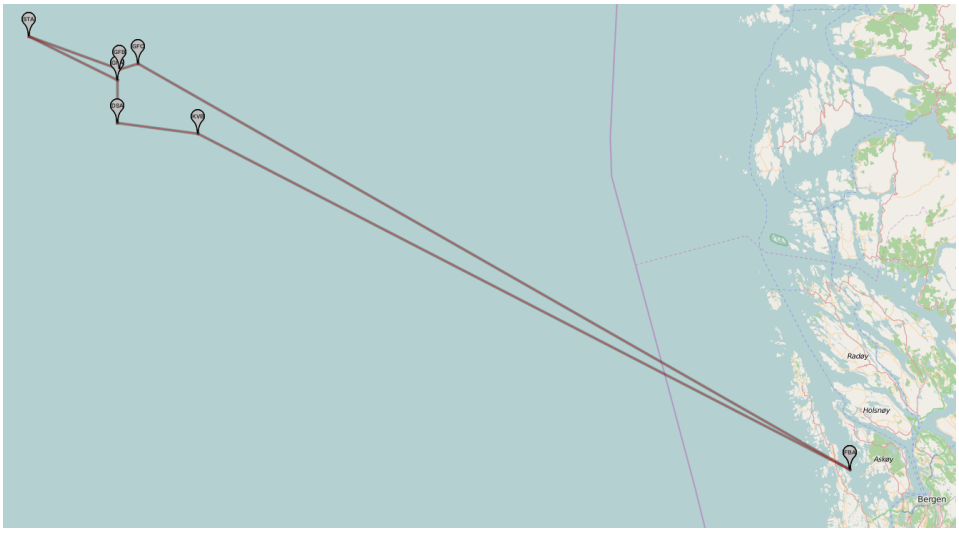

Figure D.8: ID: Å_{14}^{S}, Spot: Yes, Delay: 0 hours, Postponed: 0 nodes, Objective: NOK 1 159 260, Run time: 0.19 seconds.

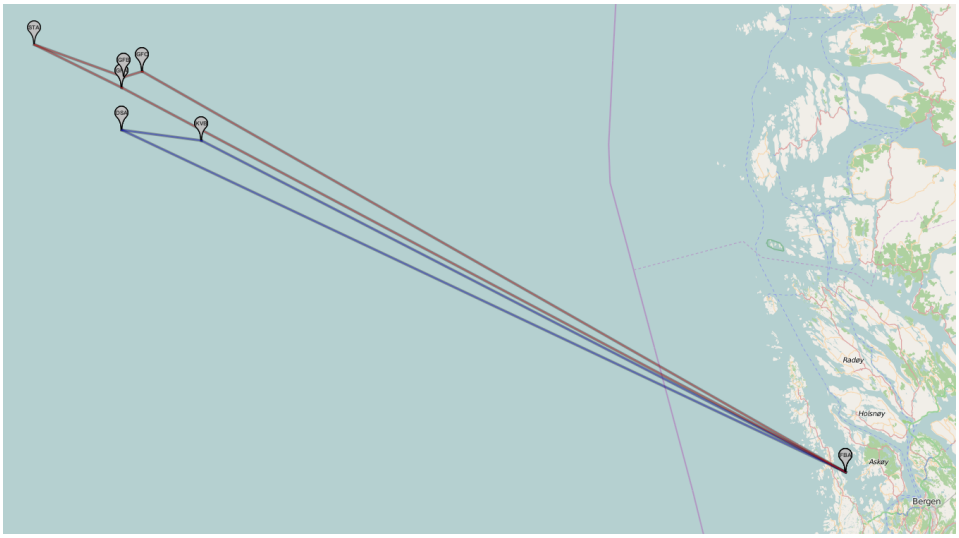

Figure D.9: ID: Å_{14}^{L}, Spot: Yes, Delay: 0 hours, Postponed: 1 node, Objective: NOK 1 084 405, Run time: 0.18 seconds.

Figure D.10: ID: F_{16}, Spot: No, Delay: 0 hours, Postponed: 0 nodes, Objective: NOK 179 569, Run time: 0.20 seconds.

Figure D.11: ID: F_{16}^{S}, Spot: Yes, Delay: 0 hours, Postponed: 0 nodes, Objective: NOK 1 152 666, Run time: 0.29 seconds.

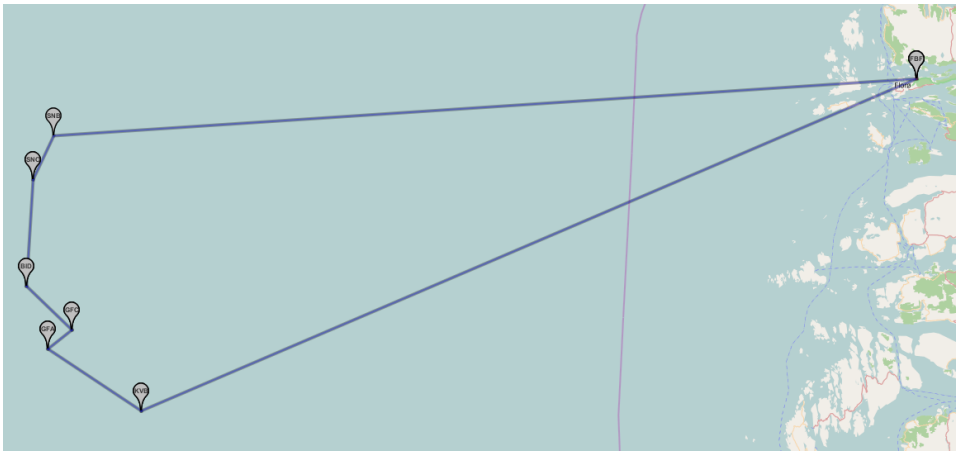

Figure D.12: ID: F_{16}^{L}, Spot: No, Delay: 0 hours, Postponed: 3 nodes, Objective: NOK 766 866, Run time: 0.19 seconds.

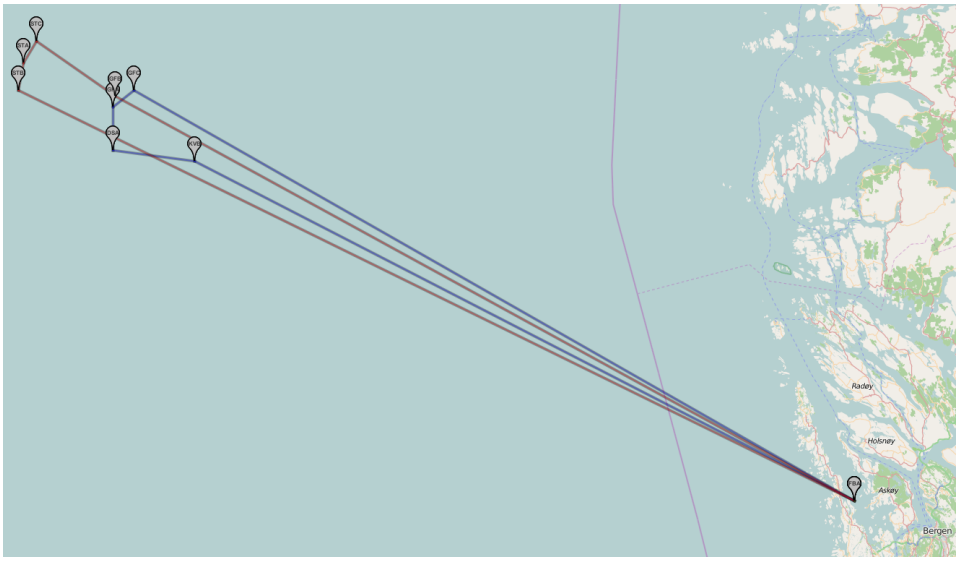

Figure D.13: ID: Å_{18}, Spot: No, Delay: 0 hours, Postponed: 0 nodes, Objective: NOK 319 480, Run time: 10.3 seconds.

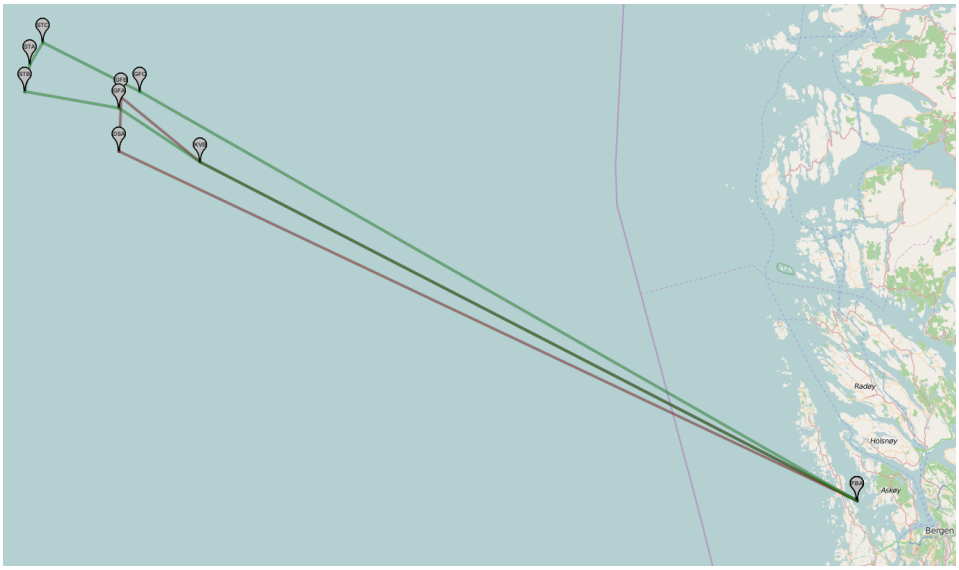

Figure D.14: ID: Å_{18}^{S}, Spot: Yes, Delay: 10 hours, Postponed: 0 nodes, Objective: NOK 1 877 761, Run time: 29.5 seconds.

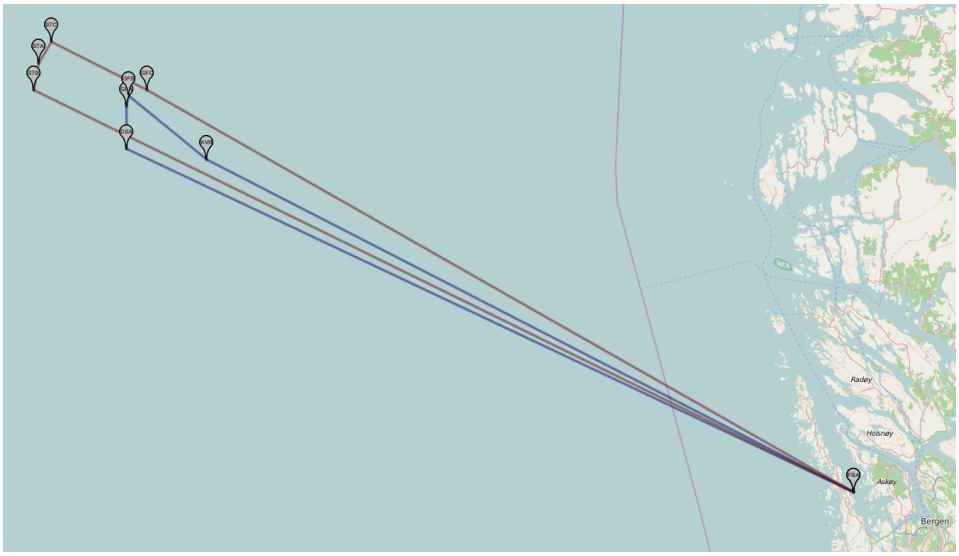

Figure D.15: ID: Å_{18}^{L} , Spot: No, Delay: 0.3 hours, Postponed: 0 nodes, Objective: NOK 321 544, Run time: 10.8 seconds.

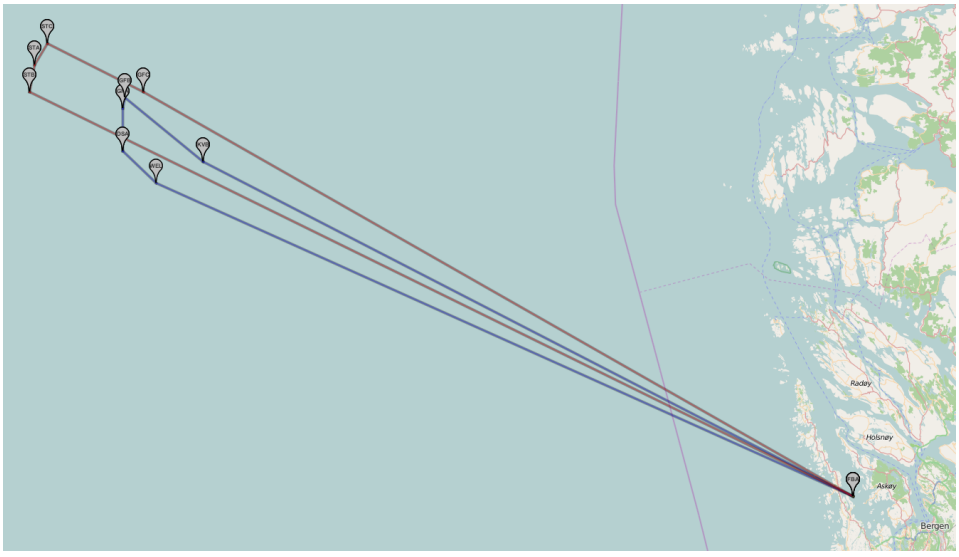

Figure D.16: ID: Å_{19}, Spot: No, Delay: 0.3 hours, Postponed: 0 nodes, Objective: NOK 325 395, Run time: 16.1 seconds.

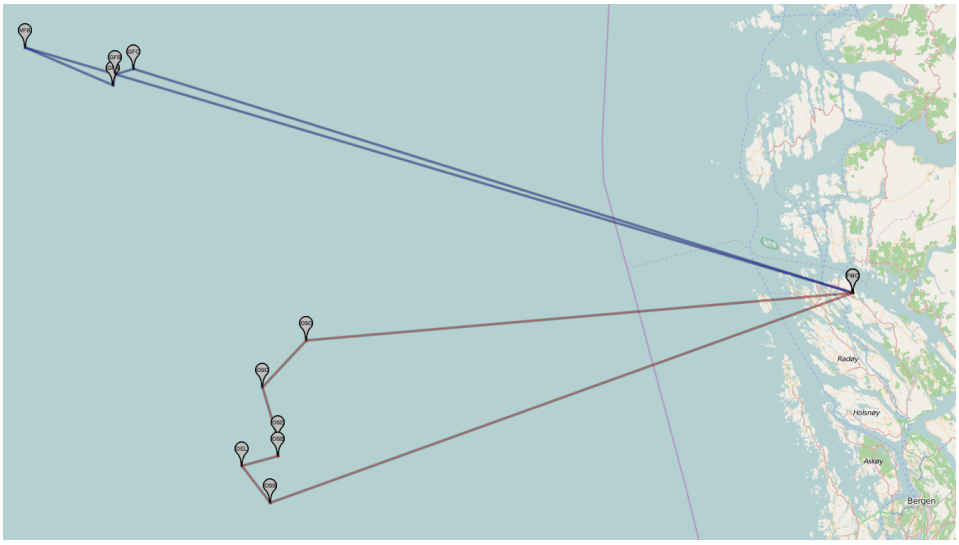

Figure D.17: ID: M_{22}, Spot: No, Delay: 0 hours, Postponed: 0 nodes, Objective: NOK 303 055, Run time: 18.4 seconds.

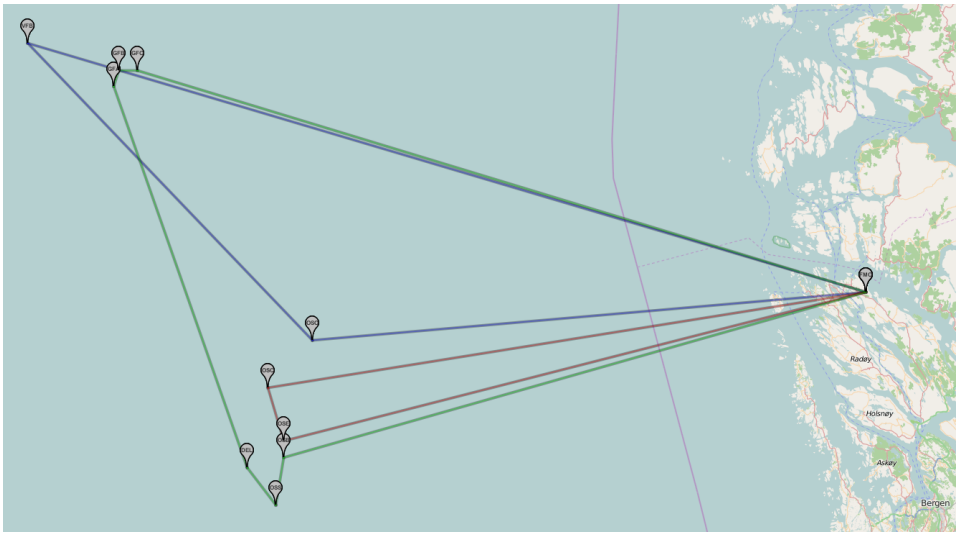

Figure D.18: ID: M_{22}^{S}, Spot: Yes, Delay: 0.3 hours, Postponed: 0 nodes, Objective: NOK 1 530 637, Run time: 28.7 seconds.

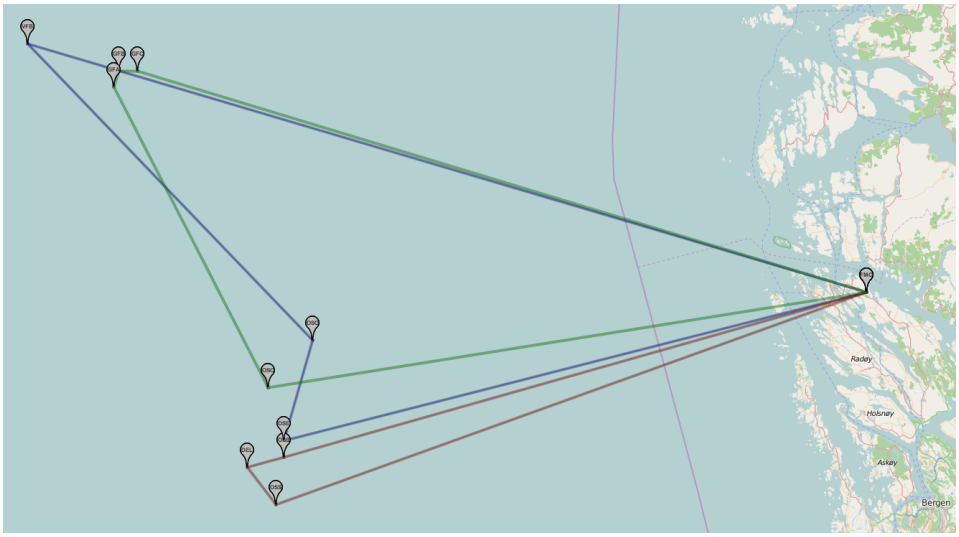

Figure D.19: ID: M_{22}^{L}, Spot: Yes, Delay: 0 hours, Postponed: 0 nodes, Objective: NOK 996 884, Run time: 18.4 seconds.

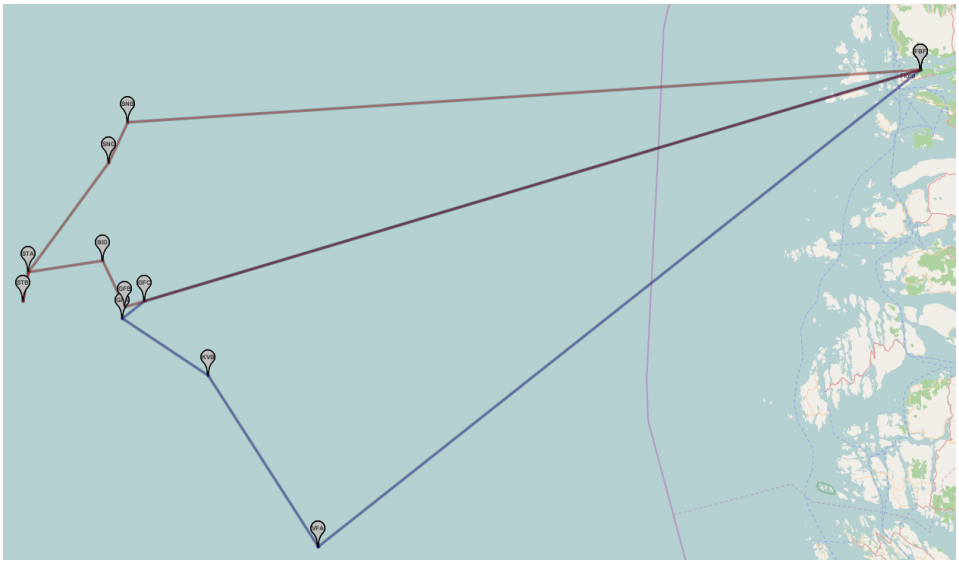

Figure D.20: ID: F_{24}, Spot: No, Delay: 0 hours, Postponed: 0 nodes, Objective: NOK 339 799, Run time: 21.5 seconds.

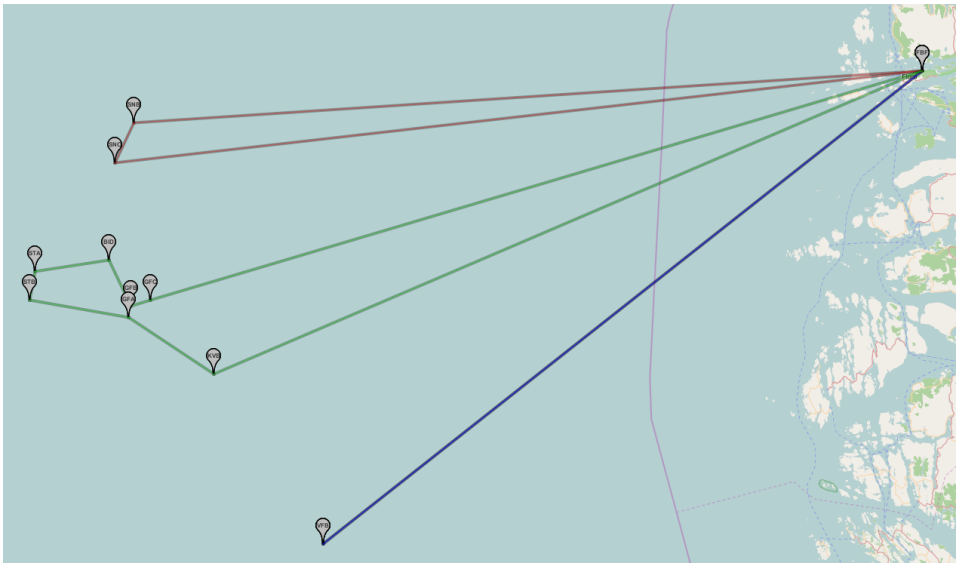

Figure D.21: ID: F_{24}^{S}, Spot: Yes, Delay: 3 hours, Postponed: 0 nodes, Objective: NOK 1 680 057, Run time: 34.4 seconds.

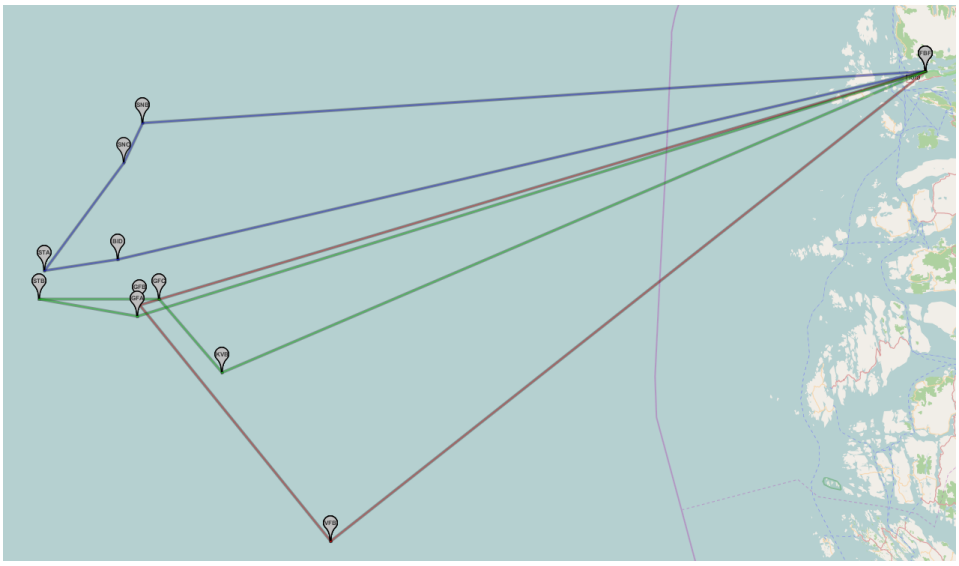

Figure D.22: ID: F_{24}^{L}, Spot: Yes, Delay: 0 hours, Postponed: 1 node, Objective: NOK 1 243 614, Run time: 59.5 seconds.

Figure D.23: ID: Å_{26}, Spot: No, Delay: 0 hours, Postponed: 0 nodes, Objective: NOK 472 051, Run time: 38.9 seconds.

Figure D.24: ID: Å_{26}^{S}, Spot: Yes, Delay: 20 hours, Postponed: 0 nodes, Objective: NOK 2 604 435, Run time: 140.5 seconds.

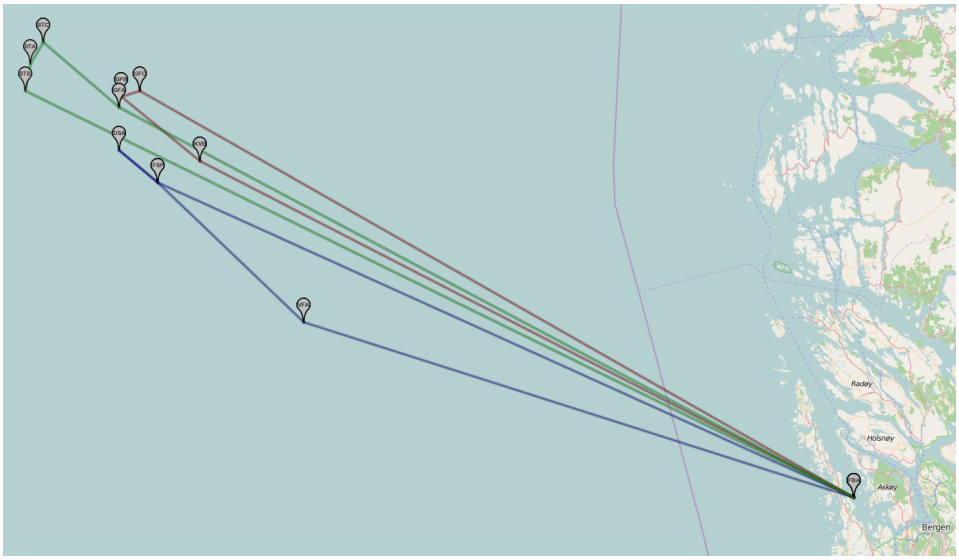

Figure D.25: ID: $\mathring{\text{A}}_{26}^{L}$, Spot: No, Delay: 0 hours, Postponed: 5 nodes, Objective: NOK 1 443 778, Run time: 95.8 seconds.

Figure D.26: ID: M_{42}, Spot: No, Delay: 0 hours, Postponed: 0 nodes, Objective: NOK 581 993, Run time: 182.2 seconds.

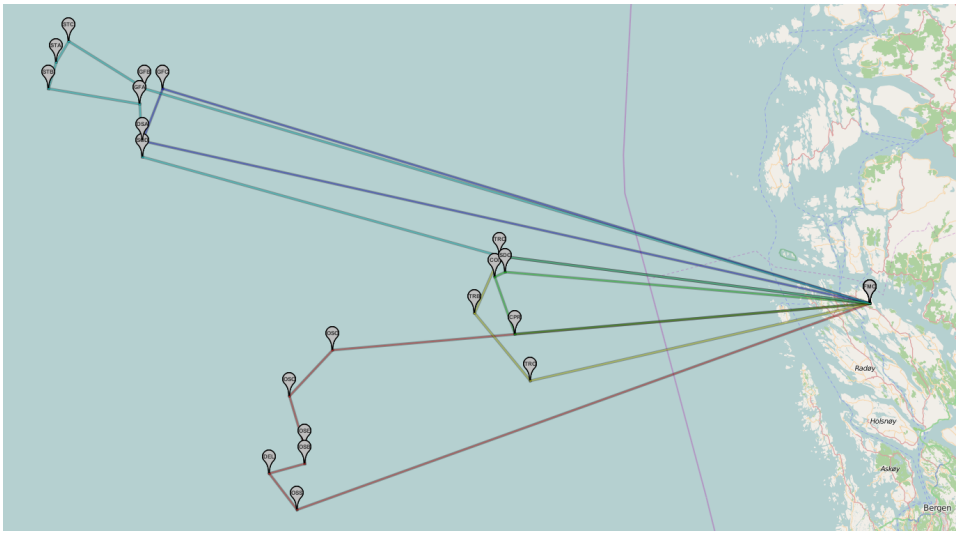

Figure D.27: ID: M_{42}^{S}, Spot: Yes, Delay: 16 hours, Postponed: 0 nodes, Objective: NOK 2 687 896, Run time: 357.5 seconds.

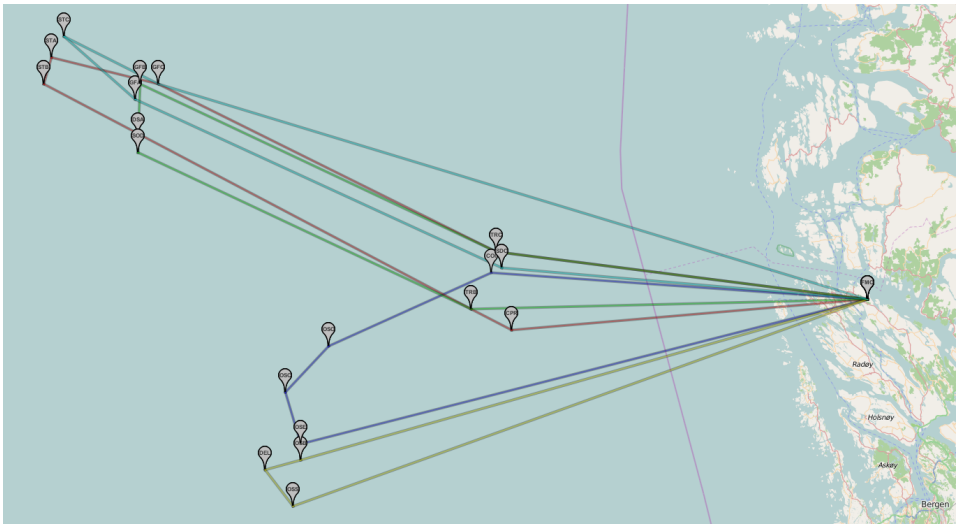

Figure D.28: ID: M_{42}^{L}, Spot: Yes, Delay: 0 hours, Postponed: 3 nodes, Objective: NOK 1 907 804, Run time: 485.4 seconds.

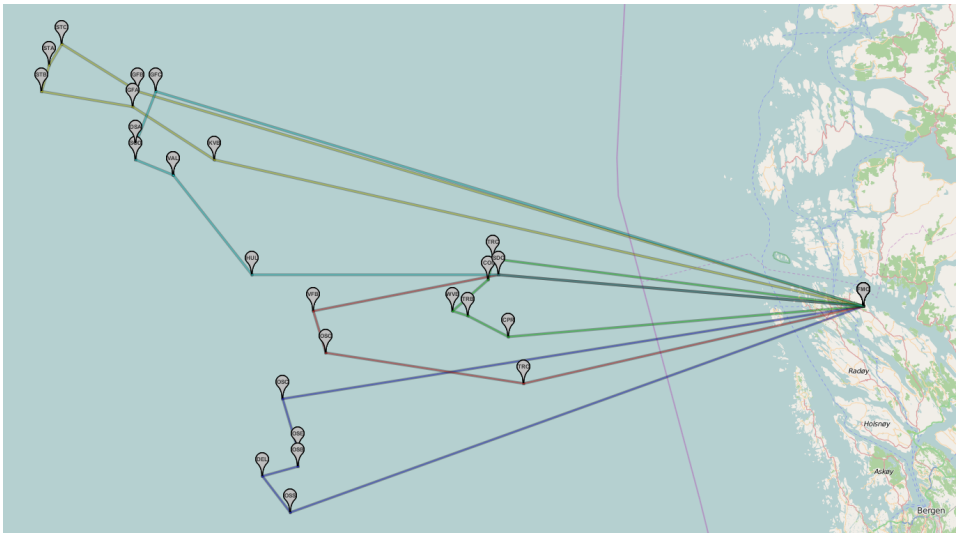

Figure D.29: ID: M_{54}, Spot: No, Delay: 0 hours, Postponed: 0 nodes, Objective: NOK 720 577, Run time: 483.8 seconds.

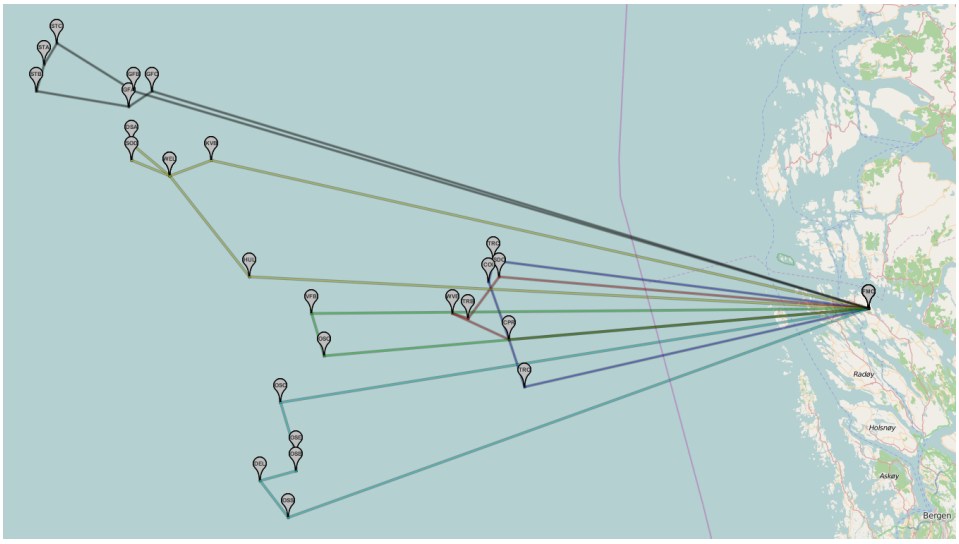

Figure D.30: ID: M_{54}^{S}, Spot: Yes, Delay: 22 hours, Postponed: 0 nodes, Objective: NOK 3 216 515, Run time: 1438 seconds.

Figure D.31: ID: M_{54}^{L}, Spot: Yes, Delay: 0 hours, Postponed: 3 nodes, Objective: NOK 2 037 586, Run time: 1602 seconds.